
TFG: Análisis Emocional para la Inclusión Digital



Gema Eugercios Suárez
Paloma Gutiérrez Merino
Elena Kaloyanova Popova

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Mayo 2018

Documento maquetado con T_EX!S v.1.0.

Este documento está preparado para ser imprimido a doble cara.

TFG: Análisis Emocional para la Inclusión Digital

Informe técnico del departamento

Ingeniería del Software e Inteligencia Artificial

IT/2009/3

Versión 1.0

**Departamento de Ingeniería del Software e Inteligencia
Artificial**

Facultad de Informática

Universidad Complutense de Madrid

Mayo 2018

Copyright © Marco Antonio y Pedro Pablo Gómez Martín

ISBN 978-84-692-7109-4

Al duque de Béjar
y
a tí, lector carísimo

*I can't go to a restaurant and
order food because I keep looking
at the fonts on the menu.
Donald Knuth*

Agradecimientos

*A todos los que la presente vieron y
entendieron.*

Inicio de las Leyes Orgánicas. Juan
Carlos I

Groucho Marx decía que encontraba a la televisión muy educativa porque cada vez que alguien la encendía, él se iba a otra habitación a leer un libro. Utilizando un esquema similar, nosotros queremos agradecer al Word de Microsoft el habernos forzado a utilizar L^AT_EX. Cualquiera que haya intentado escribir un documento de más de 150 páginas con esta aplicación entenderá a qué nos referimos. Y lo decimos porque nuestra andadura con L^AT_EX comenzó, precisamente, después de escribir un documento de algo más de 200 páginas. Una vez terminado decidimos que nunca más pasaríamos por ahí. Y entonces caímos en L^AT_EX.

Es muy posible que hubiéramos llegado al mismo sitio de todas formas, ya que en el mundo académico a la hora de escribir artículos y contribuciones a congresos lo más extendido es L^AT_EX. Sin embargo, también es cierto que cuando intentas escribir un documento grande en L^AT_EX por tu cuenta y riesgo sin un enlace del tipo “*Author instructions*”, se hace cuesta arriba, pues uno no sabe por donde empezar.

Y ahí es donde debemos agradecer tanto a Pablo Gervás como a Miguel Palomino su ayuda. El primero nos ofreció el código fuente de una programación docente que había hecho unos años atrás y que nos sirvió de inspiración (por ejemplo, el fichero `guionado.tex` de T_EX_{IS} tiene una estructura casi exacta a la suya e incluso puede que el nombre sea el mismo). El segundo nos dejó husmear en el código fuente de su propia tesis donde, además de otras cosas más interesantes pero menos curiosas, descubrimos que aún hay gente que escribe los acentos españoles con el `\’{\i}`.

No podemos tampoco olvidar a los numerosos autores de los libros y tutoriales de L^AT_EX que no sólo permiten descargar esos manuales sin coste adicional, sino que también dejan disponible el código fuente. Estamos pensando en Tobias Oetiker, Hubert Partl, Irene Hyna y Elisabeth Schlegl, autores del famoso “The Not So Short Introduction to L^AT_EX 2_ε” y en Tomás

Bautista, autor de la traducción al español. De ellos es, entre otras muchas cosas, el entorno **example** utilizado en algunos momentos en este manual.

También estamos en deuda con Joaquín Ataz López, autor del libro “Creación de ficheros L^AT_EX con GNU Emacs”. Gracias a él dejamos de lado a WinEdt y a Kile, los editores que por entonces utilizábamos en entornos Windows y Linux respectivamente, y nos pasamos a emacs. El tiempo de escritura que nos ahorramos por no mover las manos del teclado para desplazar el cursor o por no tener que escribir `\emph` una y otra vez se lo debemos a él; nuestro ocio y vida social se lo agradecen.

Por último, gracias a toda esa gente creadora de manuales, tutoriales, documentación de paquetes o respuestas en foros que hemos utilizado y seguiremos utilizando en nuestro quehacer como usuarios de L^AT_EX. Sabéis un montón.

Y para terminar, a Donal Knuth, Leslie Lamport y todos los que hacen y han hecho posible que hoy puedas estar leyendo estas líneas.

Resumen

...

...

...

Índice

Agradecimientos	IX
Resumen	XI
1. Introducción	1
1.1. Introducción	1
Notas bibliográficas	1
En el próximo capítulo	1
2. Estado del arte	3
2.1. Computación Afectiva	3
2.1.1. Diccionarios Afectivos Existentes	5
2.1.2. Nuestro diccionario	7
2.2. Servicios Web	8
2.3. Metodología Scrum	9
2.3.1. Proceso de desarrollo software	9
2.3.2. Modelo ágil: SCRUM	10
2.4. Integración Continua	13
Notas bibliográficas	14
En el próximo capítulo	15
A. Así se hizo...	17
A.1. Introducción	17
Bibliografía	19

Índice de figuras

2.1. Autoevaluación del maniquí (SAM).	7
2.2. Fragmento de la adaptación del diccionario ANEW traducido.	8
2.3. Flujo SCRUM	12

Índice de Tablas

Capítulo 1

Introducción

...

RESUMEN: Este capítulo sirve como introducción al trabajo que se va a realizar.

1.1. Introducción

...

Notas bibliográficas

Citamos algo para que aparezca en la bibliografía... (Bautista et al., 1998)

Y también ponemos el acrónimo CVS para que no cruja.

Ten en cuenta que si no quieres acrónimos (o no quieres que te falle la compilación en “release” mientras no tengas ninguno) basta con que no definas la constante `\acronimosEnRelease` (en `config.tex`).

En el próximo capítulo...

...

Capítulo 2

Estado del arte

...

...

RESUMEN: En este capítulo se van a tratar los aspectos más importantes tanto de la computación emocional como de las diferentes tecnologías y metodologías que se van a utilizar. En primer lugar, en la sección 2.1, se define la computación emocional y sus posibles aplicaciones, también se explican los distintos diccionarios afectivos ya existentes que permiten la marcación emocional de textos y se presenta el diccionario que se va a utilizar en este trabajo. En la sección 2.2 se introduce la tecnología que se va a utilizar para implementar el trabajo, los Servicios Web. En la sección 2.3 se fijan los conceptos relacionados con la metodología Scrum. En la sección 2.4 se explican las bases de la integración continua aplicada al desarrollo de Software y cómo se va a aplicar en este trabajo.

2.1. Computación Afectiva

La computación emocional es el estudio y el desarrollo de sistemas y dispositivos capaces de percibir, medir e interpretar las emociones humanas. Esta rama de la computación permite un avance notable en la inteligencia artificial, hasta tal punto que los ordenadores lleguen a adaptarse a los humanos, sus necesidades y estados de ánimo. Los seres humanos están rodeados de emociones, en cualquier ámbito de su vida, tanto de las suyas propias como las de las personas con las que se comunican. Tanta importancia tienen para nosotros que influye no sólo a nuestra comunicación, sino también en nuestro aprendizaje y toma de decisiones. Por ello, resulta artificial y en ocasiones incluso frustrante intentar comunicarse con una máquina que no es capaz de computar sentimientos. Se acuña entonces el

término computación afectiva (Picard,1997) pretende mejorar la interacción hombre-máquina haciéndola más natural y asequible.

El funcionamiento de este tipo de sistemas se basa en identificar el estado emocional del sujeto a través de diferentes fuentes(voz, expresiones, señales fisiológicas, palabras...) y procesar la información para clasificarla y aprender de ella. Clasificar la información de entrada puede resultar complicado ya que se suelen recibir varias señales diferentes a la vez, lo que hace necesario utilizar técnicas de priorización para determinar cuáles son las que más aportan a la hora de analizar y gestionar la información. Una vez identificada la emoción predominante, el sistema responde adecuándose a ella. La salida dependerá del tipo de sistema y las herramientas de las que dispone este para expresar su respuesta (colores, sonidos, emoticonos...) En casos más complejos aplicados a robótica o modelado, la salida producida es una simulación de la respuesta que produciría un ser humano ante los estímulos recibidos imitando su expresión corporal, voz o gestos faciales.

Se trata de una tecnología con un espectro enorme de posibilidades, ya que como se ha mencionado antes, las emociones están presentes en todos los ámbitos de la vida de una persona. Por lo tanto, puede aplicarse a áreas muy diferentes entre sí. Algunas de estas áreas son:

- **Marketing:** Actualmente una de sus aplicaciones más explotadas. Poder evaluar la reacción emocional de alguien ante un anuncio o producto es una estrategia comercial que ya está siendo utilizada por cada vez más empresas.
- **Salud:** Principalmente detección del estrés para minimizar sus efectos y aprender a controlarlo. Es posible inferir el nivel de estrés de una persona midiendo sus señales fisiológicas (ritmo cardiaco, respiración...) y si este nivel es demasiado alto proceder en consecuencia según el tipo de sistema. Se puede aplicar de forma similar a las fobias.
- **Entretenimiento:** La industria de los videojuegos ha crecido mucho en los últimos años e introducir este tipo de tecnología permite a las compañías crear juegos más adaptables y cercanos al jugador, lo que atrae a más público y mejora la experiencia de juego (modelado de personajes más realista y mayor empatía con estos).
- **Robótica:** El mayor problema de los robots diseñados para interactuar con humanos es la carencia de emociones. Algunos de ellos llegan a producir una sensación de incomodidad. Dotar a este tipo de robots de cierta "humanidad"no sólo haría más cómodo el tratar con ellos sino que podrían realizar tareas como el acompañamiento de personas mayores.
- **Accesibilidad:** Las herramientas que puede generar la computación afectiva pueden servir de gran ayuda a personas que no tengan la capa-

cidad de entender o expresar sus emociones, como por ejemplo personas con autismo. Para este tipo de personas sería extremadamente complicado acceder a la tecnología sin la inclusión de estas herramientas que sean capaces de adaptarse a sus necesidades.

En este trabajo vamos a centrarnos en el último área, la accesibilidad. En particular, en facilitar a personas que padecen Trastornos del Espectro Autista (TEA) el entendimiento de textos. Trataremos de analizar un texto de entrada para identificar las emociones predominantes y en que medida se presentan, evitando las posibles ambigüedades que pueda haber. Las emociones con las que trataremos son las básicas: alegría, tristeza, miedo, sorpresa, enfado o neutral. Una vez identificadas las emociones se etiquetarán para hacerlas más explícitas mediante emoticonos.

Para deducir la emoción que transmite una palabra específica utilizaremos un diccionario afectivo basado en otros ya existentes.

2.1.1. Diccionarios Afectivos Existentes

Hablamos de diccionario afectivo cuando se trata de un diccionario en el que las palabras se encuentran marcadas con etiquetas afectivas. Algunos ejemplos de etiquetas afectivas serían: alegría, tristeza, sorpresa, miedo...

Con el paso del tiempo se han ido desarrollando algunos diccionarios afectivos, los cuales describiremos a continuación:

LASSWELL VALUE DICTIONARY Y GENERAL INQUIRER

Fue a comienzos de los años 60 cuando Stone y Lasswell, ambos investigadores, comenzaron a construir diccionarios en lo que las palabras estaban marcadas con etiquetas afectivas. El diccionario Lasswell clasifica sus palabras de manera binaria y en ocho categorías básicas (riqueza, poder, respeto, rectitud, habilidad, iluminación, afecto y bienestar). Además de estas, el esquema de este diccionario distingue entre objetivos sustantivos y los elementos y atributos del proceso de distribución de valores, es decir, evaluación personal y asignación social. La clasificación social del contenido mantiene estas distinciones. Las preocupaciones con valores particulares se clasifican en objetivos sustantivos particulares. El esquema de clasificación también distingue varios tipos de transacciones de valor.

DICCIONARIO DE HATZIVASSILOGLOU Y MCKEOWN

Este diccionario fue creado en 1997, y trata de encontrar etiquetas como positivo o negativo de manera automatizada a través del análisis de un corpus. Hatzivassiloglou y McKeown tomaron una serie de adjetivos que aparecían de manera frecuente y decidieron darles una orientación, empleando un análisis estadístico en el que cuando dos adjetivos aparecían juntos, siguiendo el patrón adjetivo1 y adjetivo 2, devolvía una orientación positiva o negativa. Así, pudieron obtener un diccionario de adjetivos clasificados en positivos y negativos.

WORDNET AFFECT

Se trata de una extensión de WordNet Domains, que incluye un subconjunto de synsets adecuados para representar conceptos afectivos correlacionados con palabras afectivas. De manera similar a nuestro método para etiquetas de dominio, asignamos a una serie de sintonías de WordNet una o más etiquetas afectivas. En particular, los recursos afectivos que representan el estado emocional son individualizados por sintonizadores marcados por la emoción de la etiqueta A. También existen otras etiquetas para aquellos conceptos que representan estados de ánimo, situaciones que provocan emociones o respuestas emocionales. El recurso se amplió con un conjunto de etiquetas A adicionales (llamadas categorías emocionales), organizadas jerárquicamente, con el fin de especializar synsets con una emoción de etiqueta. La estructura jerárquica de las nuevas etiquetas se modela en las relaciones de hiperónimos de WordNet. En una segunda etapa, se definen cuatro etiquetas A-adicionales: positivo, negativo, ambiguo, neutral. Otra propiedad importante para el léxico afectivo en lo que se refiere principalmente a la interpretación de las etiquetas es la dimensión estacionaria/causativa.

WHISSELL'S DICTIONARY OF AFFECT IN LANGUAGE (DAL)

Este diccionario es un recurso diseñado con el fin de medir el significado emocional de las palabras y textos. El objetivo del diccionario es comparar las palabras con una lista de 8742 palabras que han sido marcadas por distintas personas en términos de las dimensiones activación y evaluación e imágenes.

DICCIONARIO DE TURNEY Y LITTMAN

Turney y Littman encontraron una manera más eficiente para decidir cuando una palabra se podía considerar positiva o negativa. Dado un conjunto de palabras, que previamente ellos conocían como positivas o negativas, probaron como de frecuentemente podía aparecer una palabra en un contexto con un conjunto de palabras positivas o negativas. Empleando la información de estudios cercanos y estadísticas clasificaron como positivas todas aquellas palabras que aparecían de forma más significativa con un conjunto de palabras positivas y como negativas aquellas palabras que aparecían con un conjunto de palabras negativas.

CLAIRVOYANCE AFFECT LEXICON

Este diccionario fue desarrollado a mano. Las entradas de este diccionario se dividen en cinco campos: el lema de la palabra, part-of-speech, clase afectiva, centralidad e intensidad.

ANEW Se trata de un modelo espacial de varias palabras con carga emocional que contribuye a clasificar textos arbitrarios.

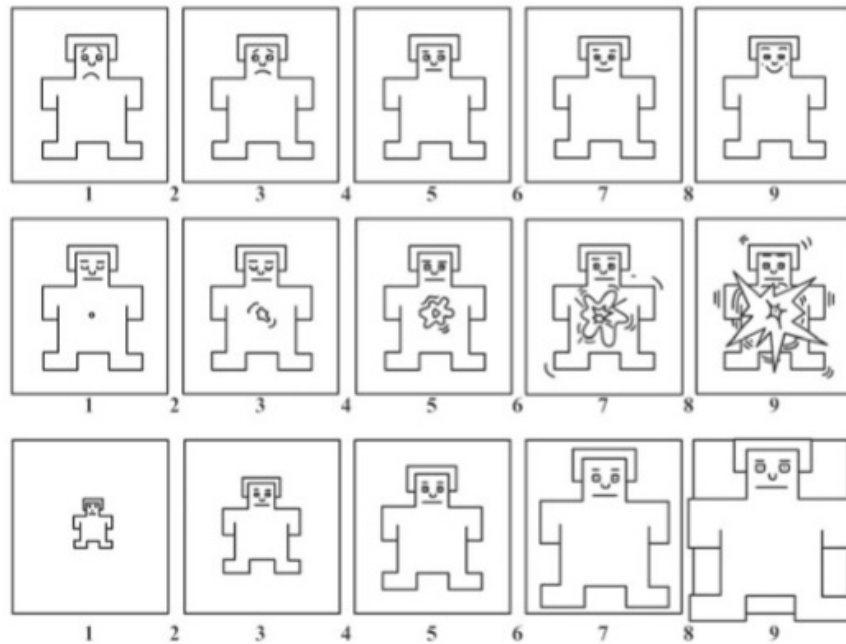


Figura 2.1: Autoevaluación del maniquí (SAM).

2.1.2. Nuestro diccionario

Nuestro diccionario afectivo está basado en el diccionario afectivo ANEW traducido al español que un equipo de la Universidad de Santiago de Compostela ha llevado a cabo (Jaime Redondo, Isabel Fraga, Isabel Padrón, y Montserrat Comesaña). Este diccionario afectivo ANEW traducido siguió un método de creación. Participaron 720 estudiantes de psicología de entre 18 y 25 años, 560 mujeres y 160 hombres. Las clasificaciones de las palabras se obtuvieron en universidades españolas, en facultades de Psicología (Santiago de Compostela, A Coruña, Valencia y Málaga) entre el 2003 y el 2005. Estos participantes deberían evaluar 1034 palabras. Estas 1034 palabras son las que contiene ANEW, que una vez traducidas y revisadas se ordenaron de manera aleatoria. Para que el trabajo no fuese tan tedioso se dividió en ocho hojas con unas 129 palabras que cada participante tendría que evaluar según las tres dimensiones emocionales del ANEW (valencia?, excitación y dominancia); estas dimensiones se podían clasificar con una escala de 9 puntos usando la autoevaluación del maniquí (SAM).

Estas evaluaciones se realizaron a la vez, en un aula. Cada participante tenía una hoja de respuestas (con 129 palabras), una hoja de instrucciones y una hoja con la SAM (Figura 2.1). Después de leer las instrucciones se resolvieron las dudas y se pusieron seis ejemplos para que los participantes entendiesen la metodología. Después comenzaron la evaluación.

S-Word	Tristeza	Miedo	Alegría	Enfado	Sorpresa	Neutral
abandonado	1,00	0,00	0,00	0,00	0,00	0,00
abejas	0,00	0,50	0,17	0,00	0,00	0,33
aborto	0,67	0,17	0,00	0,00	0,00	0,17

Figura 2.2: Fragmento de la adaptación del diccionario ANEW traducido.

Los resultados se almacenaron en una base de datos. Cada entrada en la base de datos contiene un número que indentifica a la palabra, esta numeración coincide con el número que dicha palabra tiene en el ANEW; la palabra inglesa (E-word), la palabra original en la base de datos ANEW; la palabra española (S-word), ordenadas alfabéticamente y las evaluaciones afectivas, los valores medios y la desviación estándar para cada dimensión emocional.

Basandonos en este diccionario, se ha creado una adaptación del ANEW traducido, un diccionario emocional más simplificado; ya que, aún teniendo 1034 palabras en el diccionario no se contemplan las tres dimensiones emocionales. En esta adaptación, se han elegido cinco emociones básicas (tristeza, miedo, alegría, enfado y sorpresa) y la neutralidad para no expresar ninguna emoción. Cada palabra puede representar varias de estas emociones, para ello se ha creado un número de confianza o confidence que determina con qué certeza la palabra corresponde con esa emoción. La suma de estos números de todas las emociones ha de sumar 1.

Por ejemplo en la Figura 2.2, podemos ver como la palabra abejas tiene un 0 en tristeza un 0,5 en miedo, 0,17 en alegría, 0 en enfado, 0 en sorpresa y 0,33 en neutro; lo que hace un $0,5+0,17+0,33 = 1$ de confidence. Este diccionario ha sido rellenado también con un método, menos extenso que el ANEW dado que la traducción de palabras está tomada del ANEW traducido. Para conseguir estos números de certeza (confidence) se ha repartido una hoja de cálculo a N personas, X mujeres y Y hombres, los cuales tenían que marcar para cada palabra una única emoción. Después simplemente se han unido estos datos y creado estadísticamente los números de confidence.

2.2. Servicios Web

Existen múltiples definiciones de Servicios Web por lo que resulta muy complejo dar una definición de este término. Esta complejidad nos permite hacernos una idea de la cantidad de servicios e implicaciones asociadas que se agrupan bajo este.

En la actualidad, la definición más general de Servicio Web es la que lo define como un conjunto de APIs o tecnologías capaces de interoperar en la web. Estas APIs o tecnologías intercambian datos con el fin de ofrecer unos

servicios.

También podríamos definir servicio web como un servicio ofrecido por una aplicación que expone su lógica a clientes de cualquier plataforma mediante una interfaz accesible a través de la red utilizando tecnologías estándar de internet.

Los servicios web se componen de varias capas entre las que destacan: servicios de transporte (constituidos por los protocolos del nivel más bajo, que codifican la información independientemente de su formato y que pueden ser comunes a otros servicios), servicios de mensajería, de descripción y de descubrimiento.

1. **Servicios de transporte** Es el encargado del transporte de los mensajes entre aplicaciones sobre la red. Incluye varios protocolos del nivel de aplicación. Los más utilizados son:

- HTTP: es un protocolo de nivel de aplicación más utilizado en la red. Define la sintaxis y la semántica utilizada para la arquitectura web. En relación a los servicios web es utilizado para la transferencia de las transacciones XML a través de la red utilizando los mismos principios de HTML.
- FTP: es un protocolo de la capa de aplicación encargado de los servicios de transmisión de archivos a través de redes soportadas sobre TCP. En relación a los servicios web permite realizar modificaciones en equipos remotos evitando el uso de permisos sobre los archivos en la máquina cliente en sistemas operativos que no sean Windows.
- SMTP: estándar de la capa de aplicación ampliamente utilizado para el envío de mensajes de correo electrónico a través de la red. Es un estándar que requiere cliente software de tipo POP3 o IMAP.
- BEEP: protocolo de nivel de aplicación diseñado para la interacción asíncrona punto a punto sobre una red TCP/IP. Provee un marco para administrar las conexiones punto a punto, autenticación, transporte de mensajes y manejo de errores.
- JMS: Es una aplicación de interface de programación para JAVA para el envío de mensajes entre dos o más cliente. Soporta dos modelos, el modelo punto a punto y el modelo de publicación y suscripción. Una aplicación JMS estaría compuesta por un proveedor JMS, el cual implementa las interfaces que proveen las características de administración y el control, clientes JMS, que son los componentes escritos en JAVA que producen y consumen los mensajes, objetos administradores y clientes nativos, los cuales son programas que utilizan los mensajes de la API.

2. **Servicio de mensajería** Conjunto encargado de la codificación de los mensajes en XML estándar y pueda así ser interpretado en cualquiera de los nodos de la red. Los componentes más utilizados son los siguientes:

- **REST:** Según Fielding es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la World Wide Web. Por lo tanto, se trata de un conjunto de principios para el diseño de redes, que es utilizado de manera común con el fin de definir una interfaz de transmisión sobre HTTP de manera análoga a como lo hace SOAP. Aunque REST no es un estándar como tal, posee un conjunto de estándares: HTTP, GIF, JPG, URL, XML....
- **RPC:** Tecnología de software que permite ejecutar una rutina en un equipo o segmento de red de manera remota. Es un paradigma popular para la implementación de sistemas distribuidos bajo arquitecturas cliente-servidor.
- **XML-RPC:** Protocolo de llamada remota que utiliza XML como lenguaje de codificación y HTTP como mecanismo de transporte. Protocolo sencillo, pues solo define algunos tipos de datos y comandos.
- **XML:** Se trata de uno de los lenguajes más utilizados para el intercambio de datos sobre la web. Es un lenguaje etiquetado, característica que le permite definir objetos de datos estructurados en partes bien definidas llamadas elementos. Una etiqueta es una señal realizada dentro del documento XML que delimita un segmento definido y con sentido de este documento.
- **SOAP:** Protocolo de la capa de aplicación para el intercambio de mensajes basados en XML sobre redes. Es una vía de transmisión entre un SOAP Sender y un SOAP Receiver, pero los mensajes SOAP deben interactuar con un conjunto de aplicaciones para que se pueda generar un diálogo SOAP.

3. **Servicio de descripción** El servicio web debe contar con una interfaz pública la cual es descrita por un formato llamado WSDL

- **WSDL:** es un tipo de documento XML que describe lo que hace un servicio web, donde se encuentra y la forma de ser invocado. Este provee información muy importante para los desarrolladores, este lenguaje describe el formato de los mensajes que utiliza y a cuales puede responder. Siempre un documento XML WSDL presenta los siguientes elementos: tipos, mensajes, tipos de puerto, límites, servicio....

4. **Servicio de descubrimiento** UDDI es un marco independiente de la plataforma para describir servicios, negocios e integrar servicios de negocio. La estructura de UDDI está basada sobre los servicios estándares de la web, lo que quiere decir que UDDI es accesible como otros servicios web. Es un esfuerzo de la industria iniciada en Septiembre de 2000. Los propietarios de los Servicios Web los publican en el registro UDDI. Una vez publicados se mantienen allí apuntadores a la descripción del servicio Web y del Servicio. La estructura de datos con UDDI está compuesta en cuatro partes: businessEntity, businessService, bindingTemplate, tModel[10]

Ventajas de los servicios web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedad o de las plataformas sobre las que se instalen.
- Fomentan los estándares y protocolos basados en texto, de tal manera que facilitan su comprensión y accesibilidad.
- Permiten la combinación e integración de distintos servicios y software independientemente de su ubicación.
- Permiten la integración justo-a-tiempo: el proceso de descubrimiento se ejecuta dinámicamente.
- Reducen la complejidad por medio del encapsulamiento.
- Dar una nueva vida a las aplicaciones de legado.
- Abren la puerta a nuevas oportunidades de negocio.
- Disminuyen el tiempo de desarrollo de las aplicaciones.
- Menor consumo de espacio y memoria RAM.
- Seguridad de los datos.

Inconvenientes de los servicios web

- No permiten comparar su grado de desarrollo con los estándares abiertos de computación distribuida.
- Bajo rendimiento sí se compara con otros modelos de computación distribuida.
- Como se apoyan en HTTP, puede esquivar medidas de seguridad relacionadas con el firewall.
- Ausencia de información para algunos lenguajes de programación.
- Al hacer uso de XML, mayor ancho de banda.

2.3. Metodología Scrum

2.3.1. Proceso de desarrollo software

Un proceso, es un conjunto de actividades, acciones y/o tareas que se realizan para crear un producto determinado; en nuestro caso un producto software.

El proceso de desarrollo software, como estructura general, consta de cinco grandes actividades: **comunicación, planeación, modelado, construcción y despliegue**. Estas actividades, siempre se realizan en este orden; pero dependiendo del modelo que se emplee variará el flujo; algunos flujos son: lineal, iterativo, evolutivo o paralelo.

El desarrollo de software hace unos años era algo novedoso y tan moderno que no había estrategias claras para crear productos. Por lo que, con motivo de ordenar todo el proceso se crearon diferentes modelos de proceso. Un modelo de proceso no es más que una estructura para realizar las actividades que forman un proceso.

Primeramente, se crean los modelos llamados tradicionales son procesos muy controlados y con muchas normas y políticas. Se le da mucha importancia a la arquitectura del software y se expresa mediante modelos. El cliente interactúa con el equipo de desarrollo mediante reuniones separadas en el tiempo. En la primera reunión se fija un contrato que el equipo debe seguir, lo que genera muchos problemas a la hora de desarrollar ya que el equipo tomará decisiones sin tener en cuenta lo que el cliente opinaría al respecto. Podríamos decir entonces que es un modelo impuesto externamente no solo por el contrato con el cliente sino porque está basado en normas de estándares. Estos modelos se llevan a cabo en proyectos con equipos grandes y distribuidos. Generan muchos artefactos y documentación.

Más adelante surgen los modelos de desarrollo ágiles. La agilidad en términos de software se podría definir como la respuesta efectiva al cambio; pero va más allá, esta idea lleva consigo toda una filosofía determinada en el Manifiesto por el Desarrollo Ágil. Este Manifiesto propone un nuevo modelo de proceso en el que los individuos e interacciones están por encima de procesos y herramientas; el software funcionando sobre la documentación extensiva; la colaboración con el cliente sobre la negociación contractual y la respuesta ante el cambio sobre seguir un plan. También se han redactado doce principios:

1. *La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.*
2. *Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.*

3. *Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.*
4. *Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto.*
5. *Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y apoyo que necesitan y confiarles la ejecución del trabajo.*
6. *El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.*
7. *El software funcionando es la medida principal de progreso.*
8. *Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.*
9. *La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.*
10. *La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.*
11. *Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.*
12. *A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para la continuación ajustar y perfeccionar su comportamiento en consecuencia.*

Dado que los modelos ágiles están diseñados para poder realizar cambios en los requisitos en cualquier momento y están pensados para pequeños grupos de trabajadores. Creemos muy conveniente para nuestro trabajo usar una metodología ágil.

2.3.2. Modelo ágil: SCRUM

SCRUM es una metodología ágil para gestionar el desarrollo de software. Fue definido por *Ikujiro Nonaka* e *Hiroataka Takeuchi* a principios de los 80; compararon la forma de trabajo con el avance en formación de scrum (melé en español) de los jugadores de rugby y por esta razón se llama la metodología SCRUM. *Nonaka* y *Takeuchi* caracterizan SCRUM por el protagonismo de equipos brillantes, auto-organizados y motivados que abordan el desarrollo de sistemas complejos partiendo de una visión general y solapando las fases del desarrollo. Más adelante, en 1995, *Ken Schwaber* presentó una metodología

basada en un ambiente SCRUM y usó el mismo término para definir la metodología. Después, en 2005, *Mike Cohn*, *Esther Derby* y *Ken Schwaber* organizaron la «Scrum Alliance» para difundir el marco de trabajo para el desarrollo software basado en la metodología SCRUM.

SCRUM divide el trabajo en diferentes unidades llamadas *sprints*, estos tienen una duración preestablecida de entre dos y cuatro semanas obteniendo siempre al final una versión del software con nuevas prestaciones listas para ser usadas. En cada *sprint* se ajusta la funcionalidad y se añaden nuevas prestaciones priorizando aquellas que aporten más valor.

A continuación, explicaremos algunos conceptos que son necesarios para entender la metodología.

Esta metodología, hace mucho énfasis en el «equipo de trabajo». Este equipo está formado por diferentes roles.

- **Product Owner**: representa al cliente. Este no está implicado directamente en el proyecto, pero se encarga de definir los objetivos y de garantizar que el equipo trabaja de manera adecuada. Tal y como hemos visto anteriormente, las metodologías ágiles incluyen al cliente de una manera más cercana al desarrollo y SCRUM lo hace de esta manera.
- **Scrum Master**: es el encargado de asegurar que el equipo no tiene problemas en sus tareas. Ayuda y guía al Scrum Team. Diríamos que es el encargado de que todo el proyecto salga adelante.
- **Scrum Team**: Como su nombre indica, es el equipo encargado de desarrollar y entregar el producto.

También define una serie de artefactos, en menor medida que cualquier modelo tradicional.

- **Product backlog**: es una lista realizada por el usuario en la que explica los requisitos del producto, ordenados por prioridad. A lo largo del desarrollo crece y evoluciona. Se denomina también «historias de usuario».
- **Sprint backlog**: es una lista de las tareas que debe realizar el equipo durante el sprint. Podríamos decir que son las historias de usuario que el equipo decide realizar en un sprint.
- **Incremento**: es el resultado de cada sprint.
- **Burn down chart**: es una gráfica de avance que mide la cantidad de requisitos del backlog del proyecto pendientes al comienzo de cada sprint, esta es actualizada a diario para comprobar el avance.

Por último, define una serie de eventos.

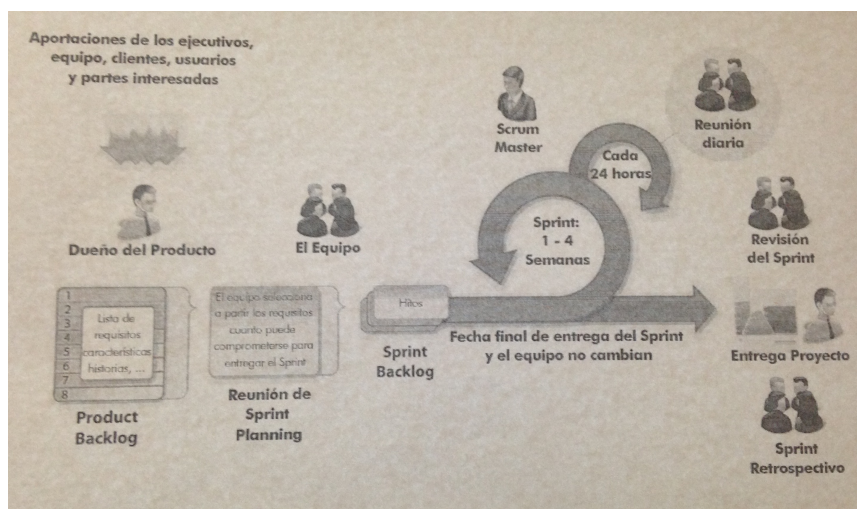


Figura 2.3: Flujo SCRUM

- **Sprint:** este evento ya está explicado. Un sprint es cada una de las iteraciones del desarrollo.
- **Sprint planning:** es una reunión del equipo de trabajo donde se decide qué se va a realizar en un sprint y las tareas en las que se divide el objetivo.
- **Dialy sprint meeting:** es una reunión diaria como máximo de quince minutos, de pie, en donde cada componente del equipo informa sobre cómo va en sus tareas, lo que hizo el día anterior, lo que hará ese día y los problemas que ha encontrado o los que cree que se va a encontrar.
- **Sprint Review:** es una reunión que se realiza al concluir el sprint donde se analiza el incremento creado, las tareas realizadas, si se ha concluido todo a tiempo, impedimentos y problemas.
- **Sprint Retrospective:** es una reunión donde se habla de cómo ha funcionado el equipo en el sprint y qué cosas se pueden mejorar para el siguiente a nivel de proceso o metodología. En esta reunión participa todo el scrum team.

Conociendo todos estos conceptos, nos resultará más sencillo comprender el flujo de trabajo.

En la Figura 2.3 podemos ver un diagrama del flujo SCRUM. Sería el siguiente: el cliente crea su Product backlog con los requisitos y características por orden de prioridades. A continuación, en un Sprint planning, se presenta el product backlog y el equipo decide qué actividades van a desarrollar y

cuánto tiempo van a tardar. Después de esta reunión se elabora el Sprint Backlog con todas las actividades que van a realizar divididas en tareas; cada componente del grupo se asigna una tarea y en cuanto acabe seguirá con la siguiente actividad que no esté hecha. Es muy importante el orden de las tareas ya que, el cliente ha ordenado en el product backlog las actividades por prioridad y el equipo desarrollará estas siguiendo dicho parámetro. A continuación se comienza el sprint con el tiempo establecido. Este tiempo no se puede cambiar ni el equipo que está trabajando. Cada día, 24 horas, el equipo se reunirá en una Daily sprint meeting para poner en común lo explicado anteriormente. Al final del sprint se realiza la Sprint Review donde se entrega el incremento o el producto hasta el momento al cliente; por lo tanto en esta reunión están presentes el scrum team, el scrum master y el cliente. Por último el scrum team junto con el scrum master se vuelve a reunir para hacer el Sprint Retrospective.

2.4. Integración Continua

La práctica de desarrollo de software que se va a utilizar en el desarrollo de este trabajo es la integración continua. Se basa en que los desarrolladores combinen todos los cambios que realicen en el código en un repositorio común de forma periódica, de tal forma que una vez subidos estos cambios, se ejecutan una serie de pruebas automáticas sobre estos con el fin de validarlos. Aparte de las pruebas automáticas se realizan pruebas manuales para buscar errores más difíciles de encontrar que requieran a alguien que piense. Hasta que las pruebas sobre el nuevo código no acaban no se puede continuar. Aplicando esto a la metodología Scrum que vamos a utilizar, una historia de usuario no se puede considerar acabada hasta que no pase todas las pruebas, tanto automáticas como manuales.

Implementar esta práctica al desarrollo tiene importantes ventajas entre las que se encuentran:

- **Detección de errores:** Su objetivo principal es detectar los errores lo antes posible y así solucionarlos según surgen. Cada vez que el código cambia se compila y somete a pruebas para garantizar que no hay bugs. Este proceso aumenta la calidad del software y minimiza los riesgos del proceso ya que se tiene control sobre las versiones en todo momento. Cuanto más se tarde en detectar un error más trabajo lleva arreglarlo.
- **Visibilidad del proceso:** Todos los pasos que se realizan en el desarrollo son visibles a todo el equipo, que tiene una estrategia común muy bien definida.
- **Mejora del equipo:** Los desarrolladores no solo tienen una visión muy clara y estructurada del proceso sino que también aprenden a realizar

todo tipo de pruebas, lo que les hace mejorar a nivel profesional.

Lo primero para poder utilizar integración continua es tener definido un «**pipeline**», es decir, un conjunto de fases por las que tiene que pasar el software y que están automatizadas. Se establecen criterios para que el código pase de una fase a otra y estrategias para gestionar errores que puedan surgir en las diferentes fases (control de versiones). Es importante tener bien definidas las pruebas que se van a realizar sobre cada fase y que estas puedan garantizar la máxima corrección posible sin tardar más de lo admisible, ya que se necesita un feedback rápido para poder seguir avanzando en el proceso. Cada fase es un grupo de pruebas y cada subida de código es un «**pipeline**» distinto que avanza de forma independiente por las fases. Por lo tanto se sabe en todo momento en qué punto se encuentra una versión específica. Esto permite tener una visión general de todo el proceso facilitando notablemente la detección de errores en fases y pipelines concretos.

Para el correcto funcionamiento de esta práctica tiene que haber pequeñas integraciones de forma frecuente, una vez al día por ejemplo. Cuantos menos cambios haya más fácil es la integración en el producto general y solucionar los posibles errores que esta pueda generar. Cabe destacar que aunque una parte de código funcione de forma independiente no implica que vaya a funcionar al integrarlo en un programa más grande, por ello cuanto más frecuentes sean las integraciones mejor.

En nuestro caso la integración continua se aplicará de la siguiente manera:

- **Repositorio:** Se utilizará un repositorio común de *GitHub* en el que se subirán todos los cambios realizados en el código. Al ser un equipo de desarrollo pequeño y estar utilizando la metodología Scrum en principio todos los miembros del equipo estarán trabajando en la misma rama. Esto puede llegar a bloquear el proceso mientras una versión acabe de pasar las pruebas.
- **Jenkins:** Tendremos Jenkins corriendo en un servidor para realizar las pruebas automáticas. Cada vez que se detecten cambios en el repositorio este avisará al servidor que procederá a hacer las pruebas de compilación y funcionamiento sobre el nuevo código.
- **Pruebas manuales:** Algún miembro del equipo realizará las tareas de «tester». Después de las pruebas automáticas se realizará otra serie de pruebas planificadas de antemano para probar mejor la funcionalidad del código. El «tester» no será siempre la misma persona.

Notas bibliográficas

Citamos algo para que aparezca en la bibliografía... (Bautista et al., 1998)

Y también ponemos el acrónimo CVS para que no cruja.

Ten en cuenta que si no quieres acrónimos (o no quieres que te falle la compilación en “release” mientras no tengas ninguno) basta con que no definas la constante `\acronimosEnRelease` (en `config.tex`).

En el próximo capítulo...

En el próximo capítulo se tratarán los primeros servicios web.

Apéndice A

Así se hizo...

...

...

RESUMEN: ...

A.1. Introducción

...

Bibliografía

*Y así, del mucho leer y del poco dormir,
se le secó el cerebro de manera que vino
a perder el juicio.*

Miguel de Cervantes Saavedra

BAUTISTA, T., OETIKER, T., PARTL, H., HYNA, I. y SCHLEGL, E. *Una Descripción de $\text{\LaTeX} 2_{\epsilon}$* . Versión electrónica, 1998.

*–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.*

*Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

*–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.*

*Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

