UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

INGENIERÍA

EN INFORMÁTICA

ANTEPROYECTO

Cuadro de Mando para la Visualización de la Calidad del Software, Integrado con un Entorno de Medición

Jose Domingo López López

Septiembre, 2010

Cuadro de Mando para la Visualización de la Calidad del Software, Integrado con un Entorno de Medición

Este documento ha sido compuesto con LaTeX. Imágenes generadas con GIMP y OpenOffice.org Draw.

Índice

1. Introducción

Años atrás, las organizaciones y empresas se preocupaban de la calidad de los procesos que se seguían para desarrollar software basándose en la premisa de que *"con un proceso de calidad se obtiene un producto de calidad"* [1], pero dados los problemas que han surgido en sistemas ya implantados y en producción, y las pérdidas que éstos han ocasionado, ha sido necesario actualizar la percepción que se tiene acerca de la calidad del software. Por ello, las organizaciones comenzaron a preocuparse por la calidad del producto software desde el punto de vista de los clientes y usuarios finales, abarcando un mayor número de *stakeholders*, con el fin de aumentar el grado de satisfacción y confianza por parte de los mismos.

Es por esto por lo que se tratado de asegurar la funcionalidad y usabilidad de los sistemas profundizando en área de *testing*, pero se han quedado sin cubrir aspectos de seguridad y mantenibilidad. Y es que un producto software no es únicamente el código que se ejecuta sobre una máquina, si no que lo son también los artefactos que se han ido produciendo a lo largo de todo el ciclo de vida: modelos, diagramas, documentos, etc.

Para paliar algunos de los problemas expuestos anteriormente, se comenzaron a utilizar dos estándares:

- **ISO/IEC 9126**[3], que define el modelo de calidad de un producto, incluyendo las características, subcaracterísticas y métricas (de calidad interna, calidad externa y calidad en uso) que han de tenerse en cuenta, pero sin proporcionar una metodología para su evaluación. Las características que propone son funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.
- **ISO/IEC 14598**[2], que proporciona guías y requisitos para el proceso de evaluación, cubriendo así los aspectos de los que carece la norma ISO/IEC 9126.

Pero entre ambos estándares existen distintas incompatibilidades e inconsistencias -tales como diferencias en el vocabulario y términos, la necesidad de añadir recomendaciones, metodologías, guías y nuevas necesidades para especificar las dimensiones de la calidad del software, entre otros- que motivan la creación de la familia **ISO/IEC 25000**, denominada SQuaRe (Software product Quality Requirements and Evaluation).

La norma ISO/IEC 25000, que consiste en 14 documentos agrupados en 5 volúmenes [14], establece criterios para la especificación de requisitos de calidad de productos software y su evaluación, un modelo de calidad, así como métricas recomendadas para los distintos atributos de dicho modelo, teniendo en cuenta a todos los *stakeholders* (clientes, usuarios finales y factorías y empresas de desarrollo).

La aparición de esta norma, cuyo objetivo es reemplazar a las normas ISO/IEC 9126 e ISO/IEC 14598, junto con las nuevas necesidades que surgen en las empresas para controlar y evaluar la calidad y seguridad de los desarrollos informáticos, motivan la aparición del proyecto **MEDUSAS** (Mejora y Evaluación del Diseño, Usabilidad, Seguridad y Mantenibilidad del Software). MEDUSAS (Fig. 1.1) es un entorno metodológico e instrumental basado en la

ISO/IEC 25000 para satisfacer dichas necesidades, y evaluar la calidad y seguridad del software teniendo en cuenta todos y cada uno de sus artefactos: código fuente, diseño, diagramas, modelos, etc.

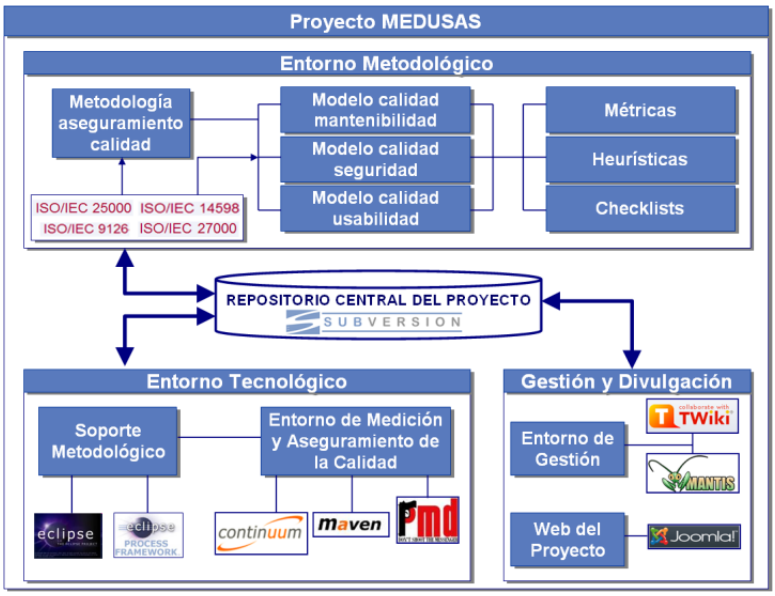

Figura 1.1: Arquitectura del proyecto MEDUSAS

Aquí es donde surge la motivación del Proyecto Fin de Carrera (en adelante PFC), que consistirá en realizar un cuadro de mando para la visualización de la calidad del software, integrado con el entorno MEDUSAS.

2. Objetivos

Los objetivos que se esperan lograr con la realización de este PFC se han dividido en dos categorías. En la primera de ellas, como objetivo principal, se introducirá el producto final que se espera obtener y, por último, como objetivos secundarios, se expondrán los objetivos que se irán alcanzando en las distintas fases de elaboración del PFC.

2.1. Objetivo principal

El PFC consiste en la elaboración de una **aplicación web** para la visualización de los datos obtenidos como resultado de las operaciones de medición y evaluación de la calidad del producto software del entorno MEDUSAS, que está basado en la nueva familia de normas ISO/IEC 25000. Para ello, se desarrollará un proyecto Java que será capaz de representar gráficamente los distintos niveles de detalle procedentes de los resultados de evaluación de la calidad del producto software.

Dicha aplicación web será desarrollada utilizando las últimas tecnologías empleadas en desarrollo y diseño web (Ver Sección 4) con el fin de obtener una RIA (Rich Internet Application) [5].

Una RIA se fundamenta en una arquitectura cliente/servidor asíncrona, segura y escalable, disminuyendo la carga de trabajo en el servidor y siendo accesible desde cualquier navegador web. Así, estará disponible para cualquier persona desde cualquier rincón del planeta, tratando de maximizar la experiencia de usuario y producir un incremento en la productividad.

2.2. Objetivos secundarios

A lo largo del proceso de investigación y elaboración del PFC se alcanzarán las siguientes metas:

- Comprender las actuales normas sobre la calidad del software.
- Investigar sobre las principales formas de visualización de calidad del software.
- Estudiar los principales entornos y medios actuales para la visualización de la calidad del software, así como las librerías gráficas que permiten su implementación.
- Implementar un entorno que permita visualizar la calidad de un producto software de acuerdo a las características de calidad impuestas por la ISO/IEC 25000.
- Integrar dicho entorno de visualización con un entorno real de medición de la calidad del producto software.

3. Método y fases de trabajo

Debido al carácter relativamente investigador que posee este proyecto, en el que se producirán numerosas adiciones y modificaciones de requisitos de usuario a lo largo de su ciclo de vida, se ha optado por utilizar una metodología de trabajo genérica que permita adaptarse a este caso de estudio. Por ello, la metodología elegida es el **Proceso Unificado de Desarrollo** (en adelante PUD).

El PUD [4] es una evolución del Proceso Unificado de Rational, que define un "conjunto de actividades necesarias para transformar los requisitos de usuario en un sistema software". Es un marco genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. Sus principales características son:

- **Dirigido por casos de uso**. Para poder desarrollar un sistema es necesario saber qué necesitan sus usuarios. Un usuario, en semejanza al concepto de *actor* en UML, puede ser un ser humano u otro sistema que interacciona con el sistema que se está desarrollando. Las necesidades de un usuario se denominan requisitos funcionales y se representan por medio de casos de uso. Por tanto, un caso de uso representa un requisito funcional del sistema que proporciona un resultado de valor a un usuario. Además, guían el proceso de desarrollo desde la especificación de requisitos hasta las pruebas y sirven a los desarrolladores para crear una serie de modelos que les permita llevarlos a cabo. Todos los casos de uso juntos constituyen el **modelo de casos de uso**.
- **Centrado en la arquitectura**. Un sistema software puede contemplarse desde varios puntos de vista. Por tanto, la arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema y debe estar profundamente relacionada con los casos de uso ya que debe permitir el desarrollo de los mismo. Tanto la arquitectura como los casos de uso deben desarrollarse en paralelo.
- **Iterativo e incremental**. Partiendo del dicho latino *divide et impera* (en castellano, *divide y vencerás*), la filosofía del PUD se basa en la creación de **mini-proyectos** para repartir el trabajo. Cada mini-proyecto es una **iteración** que resulta en un **incremento**. Las iteraciones deben ser seleccionadas y ejecutarse de forma controlada y siguen el esquema *requisitos, análisis, diseño, implementación y pruebas*, al cual denominaremos **flujo de trabajo**. Cada iteración trata un grupo de casos de uso que juntos amplían la utilidad del producto desarrollado y trata los riesgos más importantes. Si la iteración cumple sus objetivos, se continúa con la próxima.
  - *Análisis*. Consiste en identificar y especificar los casos de uso relevantes.
  - *Diseño*. Se debe crear un diseño utilizando la arquitectura seleccionada como guía para tratar de dotar al sistema de las funcionalidades representadas por los casos de uso identificados en la etapa de análisis.
  - *Implementación*. En esta estapa se llevarán a cabo, a nivel de código, las decisiones tomadas en la etapa de diseño.

- *Pruebas*. Se verificará que los cambios realizados en el sistema satisfacen los casos de uso.

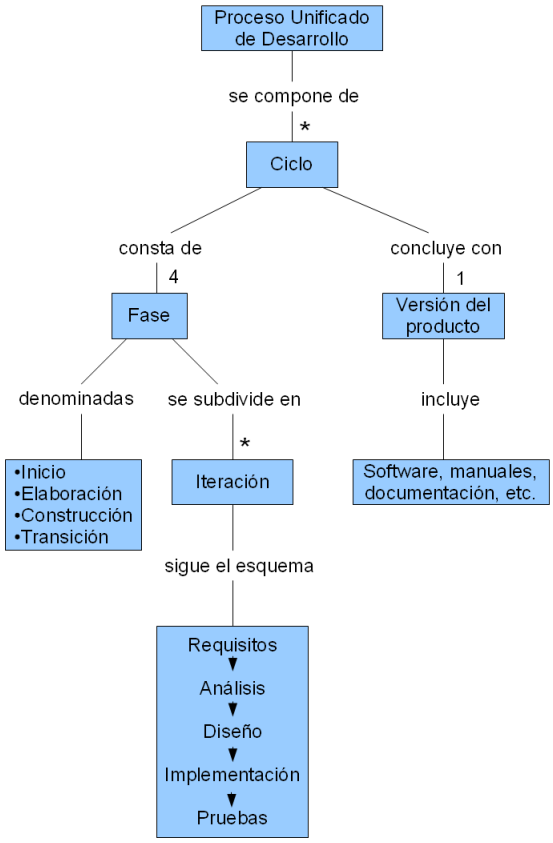

Figura 3.1: Mapa conceptual del PUD

Debido al enfoque iterativo e incremental que caracteriza al PUD, éste se divide en ciclos. Cada ciclo consta de cuatro fases que a su vez se dividen en iteraciones. Por último, cada iteración sigue el esquema *requisitos, análisis, diseño, implementación y pruebas* (Fig. 3.1).

Las cuatro fases que constituyen un ciclo son las siguientes:

- **Inicio o Factibilidad**. Se obtiene un modelo de casos de uso simplificado, se identifican riesgos potenciales y se estima el proyecto de manera aproximada.
- **Elaboración**. Se mejora el modelo de casos de uso y se diseña la arquitectura del sistema. A continuación, se desarrollan los casos de uso más críticos que se identificaron en la fase de inicio y se obtiene una *línea base* de la arquitectura para planificar las actividades y estimar los recursos necesarios para terminar el proyecto.
- **Construcción**. Se crea el producto en base a las dos fases anteriores.
- **Transición**. Implica la corrección de errores y *bugs*, así como el mantenimiento del sistema.

4. Medios que se pretenden utilizar

Como ya se ha explicado anteriormente (Ver Sección 2.1), se elaborará una aplicación web utilizando las últimas tecnologías empleadas en desarrollo y diseño web con el fin de obtener una RIA.

Una aplicación web tiene dos contextos bien diferenciados: el lado del cliente (client-side) y el lado del servidor (server-side). Para poder dotar una aplicación web de las características de una aplicación de escritorio es necesario trabajar desde el lado del cliente. Para ello, se dispone de tecnologías como CSS, Ajax y JavaScript que nos permiten cambiar dinámicamente el contenido que el usuario recibe en su navegador web a través del DOM[1].

A continuación se expondrán este tipo tecnologías así como un conjunto de librerías que serán de gran utilidad a la hora de generar diagramas y gráficas.

4.1. Tecnologías para desarrollo y diseño web

4.1.1. HTML 5

HTML 5 (HyperText Markup Language, versión 5) es la quinta revisión de HTML. Esta nueva versión incorpora de forma nativa, es decir, sin depender de programas externos o *plugins* del navegador, reproducción de audio y vídeo, edición de documentos, *drag&drop* y un elemento *canvas* sobre el que se pueden pintar gráficos en 2 y 3 dimensiones, entre otros.

4.1.2. Struts 2

Struts 2 es la nueva versión del *framework* de desarrollo web Java Apache Struts. Struts 2 está basado en el patrón MVC (Modelo-Vista-Controlador), una arquitectura que busca reducir el acoplamiento dividiendo las responsabilidades en 3 capas bien diferenciadas:

- El modelo, que hace referencia a los datos que maneja la aplicación y las reglas de negocio que operan sobre ellos.
- La vista, encargada de generar la interfaz con la que la aplicación interacciona con el usuario.
- El controlador, que comunica la vista y el modelo respondiendo a eventos generados por el usuario en la vista, invocando cambios en el modelo, y devolviendo a la vista la información del modelo necesaria para que pueda generar la respuesta adecuada para el usuario.

[1]DOM, acrónimo de Document Object Model, es un API a través del cuál se puede modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML.

No obstante, también es posible diseñar una aplicación web mediante el patrón MVC utilizando páginas JSP y servlets. En este caso, las páginas JSP tomarían el rol de la vista, los servlets el del controlador y los POJO (Plan Old Java Object) el del modelo. El modelo, a su vez, suele estar dividido en dos subcapas siguiendo el patrón DAO (Data Access Object).

En la aplicación web, Struts 2 se empleará en el desarrollo de determinados casos de uso extraídos de la especificación de requisitos de usuario, identificando dichos casos de uso como *acciones* de Struts.

En [13] se encuentra una lista con todos los plugins de los que dispone este framework: jQuery, Hibernate, jFree Chart, JUnit, YUI, GWT, etc.

4.1.3. DWR: Direct Web Remoting

DWR [7], bajo el eslogan de "Easy Ajax for Java", es una solución muy utilizada en el mundo empresarial para generar código JavaScript que, ejecutándose en el cliente, permite que los navegadores web hagan peticiones a clases Java que se ejecutan en el servidor como si éstas se ejecutasen en local. Además, con *Reverse Ajax*, es posible que el código Java que se está ejecutando en el servidor utilice un API para modificar el contenido y estilo de los clientes, estableciendo así comunicación en ambos sentidos: del cliente al servidor y del servidor al cliente. Además, es posible integrarlo con tecnologías como jQuery (client-side) y Struts e Hibernate (server-side).

4.1.4. jQuery

jQuery [10] es un *framework* para JavaScript, cuya arquitectura se basa en pequeños *plugins* muy específicos. Este framework está muy extendido entre la comunidad de desarrolladores de aplicaciones web y contiene grandes fuentes de documentación.

Cabe destacar **jQuery UI** [11], que se ejecuta sobre jQuery y provee al programador de un mayor nivel de abstracción en cuanto a interacciones con la interfaz gráfica de usuario, animaciones, *widgets* complejos, etc. permitiendo diseñar una aplicación web como si se tratase de una aplicación de escritorio.

4.2. Tecnologías para visualización

Una vez que se han investigado numerosas tecnologías y librerías de visualización [6], se ha seleccionado un conjunto de las mismas (que puede variar en función de futuros requisitos de usuario, es decir, no se descarta la supresión o adición de otras librerías) en base a las siguientes características:

- Generación dinámica de diagramas.

- Posibilidad de elección entre una gran colección de modelos (diagramas de barras, sectores, radar, etc.).
- Permitir la interacción por parte del usuario.
- Visualización atractiva.
- Fácil integración en la aplicación web, desarrollada en Java.
- Gratuito y, preferiblemente, libre.
- Una licencia conforme a nuestros requisitos.

4.2.1. Google Visualization API

Es una librería basada en JavaScript y perteneciente a Google [8], que permite la generación de multitud de gráficas y diagramas. Goza de un buen soporte por parte del equipo de desarrollo y una comunidad de desarrolladores que aportan una gran cantidad documentación y nuevos modelos. Entre sus características destacan las siguientes:

- Soporta interacción por parte del usuario gracias a una fuerte captura y manipulación de eventos.
- Permite ordenar, filtrar y manipular los datos.
- Los orígenes de datos pueden ser la propia página o páginas externas, por ejemplo Google Docs.
- Proporciona una gran variedad de modelos: líneas, barras verticales y horizontales, sectores, áreas, árboles, geo-mapas (ver Figura 4.1), mapas interactivos que utilizan el API de Google Maps, tablas, nubes de palabras, etc.
- El código de visualización no puede ser almacenado localmente. Se debe acceder a *http://www.google.com/jsapi* para utilizar las visualizaciones.

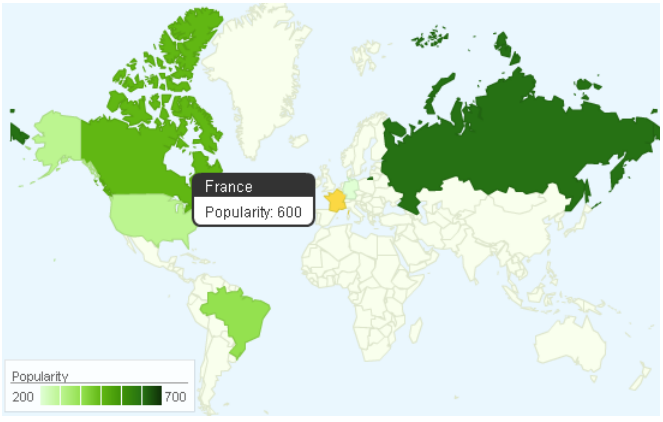

Figura 4.1: *Geomapa* de Google Visualization API

4.2.2. The JavaScript InfoVis Toolkit

Se trata de una librería [9] con licencia Creative Commons v3, desarrollada por completo en JavaScript con un gran potencial para representar y manejar grafos, *treemaps*, y *sunbursts*. Sus principales características son las siguientes:

- Soporta la captura y manipulación de eventos.
- Soporta zoom y animaciones.
- Permite que el usuario mueva los diagramas o sus elementos.
- Soporte para dispositivos móviles.
- Proporciona una gran variedad de modelos: líneas, barras verticales y horizontales, sectores, áreas, árboles. **No soporta diagramas de tipo radar**.

4.2.3. Open Flash Chart

Open Flash Chart [12] es una librería libre y gratuita, con licencia LGPL, que se basa en flash para generar los diagramas, lo cual ha sufrido algunas mejoras significativas gracias a las nuevas etiquetas que incorpora HTML5. Además, incluye un plug-in para ser integrado con numerosos lenguajes, entre ellos Java y el framework Struts 2. Entre sus características destacan las siguientes:

- Captura y manipulación de eventos.
- Grandes factores de configuración y personalización: capturar eventos, barras de carga, menús (ver Figura 4.3), tooltips, etc.
- Proporciona una gran variedad de modelos: líneas, barras verticales y horizontales, sectores, áreas, **radar** (ver Figura 4.2), etc.
- Los diagramas pueden ser exportados a imágenes estáticas.

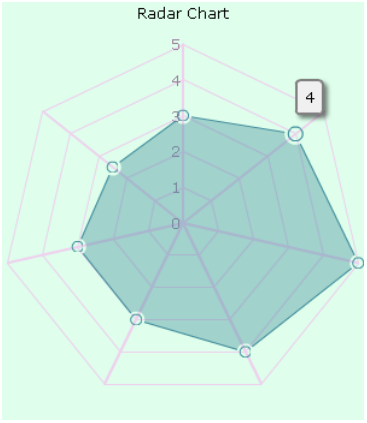

Figura 4.2: Diagrama de tipo "radar" con Open Flash Chart

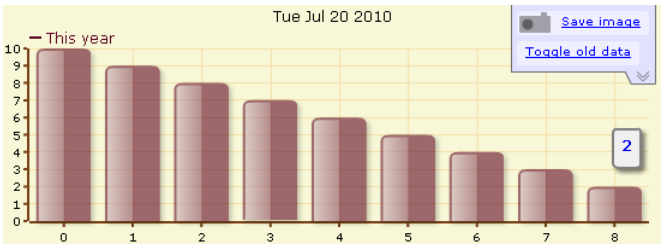

Figura 4.3: Diagrama con menú con Open Flash Chart

4.3. Otras tecnologías

4.3.1. Hibernate

Hibernate es un *framework* de persistencia, libre y bajo licencia LGPL, que permite establecer un *mapeo objeto-relacional* para la plataforma Java y sistemas basados en SQL, es decir, permite relacionar una clase Java con una tabla de una base de datos para abstraer al programador del lenguaje SQL y haciendo que la aplicación sea portable entre cualquier sistema gestor de bases de datos compatible con SQL.

Este framework, además de utilizarse para gestionar la persistencia del entorno MEDUSAS, se utilizará para gestionar la persistencia de la aplicación web: gestión de usuarios, estadísticas de uso, historiales, etc.

Referencias

[1] J. Verdugo M. Piattini D. Mellado, M. Roríguez and E. Fernández-Medina. Evaluación de la Calidad y Seguridad en Productos Software.

[2] ISO. ISO/IEC 14598 Software engineering - Product evaluation.

[3] ISO. ISO/IEC 9126 Software engineering - Product quality. 1991.

[4] I. Jacobson J. Rumbaugh and G. Booch. *El Proceso Unificado de Desarrollo de Software*. Addison-Wesley, 2000.

[5] C. Nascimbene. ¿Qué son las Rich Internet Applications? 12 2005.

[6] 22 Awesome Visualizatin Libraries: Charts and Diagrams. http://www.onextrapixel.com/2009/11/24/22-awesome-visualization-libraries-charts-and-diagrams/.

[7] Página Web de DWR: Easy Ajax for Java. http://directwebremoting.org/dwr/index.html.

[8] Página Web de Google Visualization API. http://code.google.com/intl/es-ES/apis/visualization/interactive_charts.html.

[9] Página Web de JavaScript InfoVis Toolkit. http://thejit.org/.

[10] Página Web de jQuery. http://jquery.com/.

[11] Página Web de jQuery UI. http://jqueryui.com/.

[12] Página Web de Open Flash Chart v2. http://teethgrinder.co.uk/open-flash-chart-2/.

[13] Apache Struts 2 Plugin Registry. https://cwiki.apache.org/S2PLUGINS/home.html.

[14] W. Suryn and A. Abran. ISO/IEC SQuaRE. The second generation of standards for software product quality.

Anexos

A. Contrato de Propiedad Intelectual

En Ciudad Real, a 29 de septiembre de 2010

REUNIDOS

DE UNA PARTE, **D. Jose Domingo López López** mayor de edad, con D.N.I. número **71219116-f**, Alumno de la Escuela Superior de Informática, y en adelante el **"ALUMNO DEL PFC"**.

DE OTRA PARTE, **Dña. María de los Ángeles Moraga de la Rubia** mayor de edad, con D.N.I. número **5683662-v**, Profesora de la Escuela Superior de Informática, y en adelante el **"TUTOR ACADÉMICO DEL PFC"**.

DE OTRA PARTE, **D. Moisés Rodriguez Monje** mayor de edad, con D.N.I. número **5928188-F**, Director Técnico de la empresa Alarcos Quality Center (en adelante AQC), y en adelante el **"DIRECTOR DEL PFC"**.

Todos ellos reconociéndose mutuamente capacidad suficiente para la celebración del presente Contrato,

EXPONEN

PRIMERO: El objeto del presente contrato es la determinación de la propiedad intelectual del PFC titulado **"Cuadro de Mano para la Visualización de la Calidad del Software, Integrado con un Entorno de Medición"**, y del resultado de la integración del mismo con las herramientas usadas por **AQC**. Así como, establecer el compromiso de la confidencialidad que el **ALUMNO DEL PFC** contraerá con **AQC**.

SEGUNDO: La propiedad intelectual del producto software resultante del **PFC** integrado con las herramientas y entornos de la empresa **AQC** pertenecerá única y exclusivamente a **AQC**, no pudiendo el **ALUMNO DEL PFC** utilizarlas sin el consentimiento expreso de la empresa. Sin embargo, la propiedad intelectual del **PFC** de manera aislada, así como de los documentos, diagramas, modelos, etc. que lo acompañen será compartida a partes iguales entre el **ALUMNO DEL PFC**, el **TUTOR DEL PFC** y el **DIRECTOR DEL PFC**.

TERCERO: El **ALUMNO DEL PFC** trabajará para la realización de su **PFC** con documentación, productos software y entornos de la empresa **AQC** y por ello, por medio de la presente, se compromete de forma irrevocable ante **AQC** a:

1. Asumir las políticas, los objetivos y las directrices establecidas por **AQC**, y particularmente la política de calidad, las directrices y procedimientos incluidos o referenciados en el manual de calidad de **AQC**.

2. Actuar de forma que se preserve una imagen positiva y eficad de **AQC** guardando al debida diligencia en relación con todas las labores efectuadas, actuando en todo momento de buena fe y en aras al leal cumplimiento de mis obligaciones para con **AQC**.

3. Evitar intervenir en cualquier actividad que pueda disminuir la confianza en la competencia, imparcialidad, juicio o integridad operativa de **AQC**.

4. Asegurar la imparcialidad y estar libre de toda presión indebida, comercial, financiera o de otra índole, que pueda influir en el juicio técnico empleado en la realización de sus actividades en **AQC**.

5. No alegar su calidad de colaborador de **AQC** con fines ajenos a las misiones que se le hayan confiado por **AQC**. No admitir ningún tipo de retribución ajena a **AQC** por cualquier actividad realizada en nombre de **AQC** o en calidad de persona cualificada por **AQC**.

6. En particular, durante la realización de las actividades del **PFC**:

   - Detectar desviaciones / no conformidades basadas en evidencias objetivas y no en opiniones pesonales o valoraciones subjetivas.
   - No adoptar nunca actitudes de prepotencia ante la empresa.
   - No tener inconveniente en modificar una opinión si la empresa le demuestra que estaba equivocado.
   - No ceder ante presiones o coacciones que pueda recibir si está convencido de sus criterios y no le demuestran con contrario.

7. Participar, en la medida de los posible, en las actividades, reuniones o cursos de armonización de criterios siempre y cuando sea requerido por **AQC**. Compartir con el personal de **AQC**, su conocimiento, experiencia, ideas y sugerencias.

8. Avisar a **AQC** de cualquier cambio o interrupción en sus actividades profesionales.

9. Durante su relación con **AQC** y tras ella, guardar el más estricto secreto sobre todos los asuntos, documentos y registros confidenciales de carácter técnico, organizativo, comercial y económico de **AQC** y aquella que pudiera tener conocimiento dentro de marco de sus actividades. Tratando como documentación confidencial:

   - Solicitud, formal o informal, de un proyecto. Queda incluido el hecho de la existencia de la solicitud.
   - La entrega por una organización que ha solicitado o ha realizado un proyecto con **AQC**.

- Los datos de los productos o servicios o de las organizaciones que ha solicitado o ha realizado un proyecto con **AQC**.
- Toda la correspondencia entre **AQC** y las organizaciones con que trabaja o tiene relación.
- Toda la correspondencia entre **AQC** y sus subcontratistas y viceversa.
- La que los documentos del sistema señalen como confidencial.
- Los documentos del sistema de gestión de **AQC**.
- Todos los documentos que **AQC** le facilite para la realización exitosa de su **PFC**, así como las herramientas y modelos con los que el **ALUMNO DEL PFC** deba integrar su trabajo.

10. Adoptar todas las precauciones para evitar que se divulguen, directa o indirectamente, por causa propia o de las personas de las que es responsable, documentos o informaciones de los que pudiera tener conocimiento dentro del marco de sus actividades en los proyectos de **AQC**. Estas precauciones incluyen:
    - Manipular y mantener debidamente archivada la documentación.
    - No dejar incontrolada documentación confidencial en salas de reuniones y visitas.
    - Evitar que personas ajenas accedan a lugares de trabajo y archivos.
11. En caso de acceso y tratamiento de datos de carácter personal, por exigencias el trabajo que desempeñe, asegurará la confidencialidad de los mismos respetando y cumpliendo en todo momento los procedimientos y directrices instaurados en **AQC** al efecto y la legislación vigente aplicable.
12. Respetar la titularidad de **AQC** sobre cualquier proyecto, informe, documento e imágenes que realice en el desempeño de las tareas que se ele encomiendan durante el **PFC** (respetando y protegiendo los derechos de propiedad intelectual o industrial de **AQC**).

Y en prueba de cuando antecede, las Partes suscriben el Contrato, en cuatro ejemplares y a un solo efecto, en el lugar y fecha señalados en el encabezamiento.

EL ALUMNO DEL PFC **EL TUTOR DEL PFC** **EL DIRECTOR DEL PFC**