Trabajo Final de Unidad - UT1

Para el trabajo se seleccionó el desarrollo del Gestor de Retos; el cual fue un desafío planteado en el Reto 2024 (febrero).

Parte 1

Lo primero que se debe hacer es definir el alcance. En esta etapa se deben identificar todos los *stakeholders*, no solo los clave, y determinar su nivel de involucramiento en el proyecto. Una buena herramienta para esto es realizar una matriz RACI, que clasifica a los *stakeholders* dependiendo del rol que cumplen.

Para este caso, los *stakeholders* son la Universidad Católica del Uruguay, la Facultad de Ingeniería y Tecnologías, los profesores responsables del Reto, los alumnos que desarrollan el proyecto, y todos los que en un futuro estén involucrados en los Retos, tanto profesores como alumnos. Los profesores y alumnos serían los *stakeholders* clave.

El proyecto elegido para el estudio de este informe fue sobre una aplicación creada para los docentes de la UCU en el Reto desarrollado en febrero de 2024; el Gestor de Retos. Este es un sistema cuya función principal era encargarse de que los grupos de los retos se armasen de forma automatizada.

La necesidad de dicho sistema nace de los crecientes ingresos que se están dando en la carrera de Informática en la UCU y, por ende, del creciente número de estudiantes que deben cursar Retos, pues la asignación de los alumnos a sus respectivos proyectos y grupos era, hasta la fecha, realizada manualmente por los profesores. Esto, a medida que avanza el tiempo y aumentan las inscripciones a los retos, implica una tarea cada vez mayor, consumiendo más tiempo de los profesores y dando lugar a que se cometan errores.

Al elegir cómo obtener la información de los requisitos, se utilizaron varias técnicas de elicitación. La más destacada o usada como tal fue la de las entrevistas, ya que se podía consultar directamente al cliente y principal interesado. Esta técnica dio la pauta de cómo era el flujo del proceso de forma manual, junto con las principales funcionalidades esperadas en el sistema final.

Otra técnica de elicitación de requerimientos que fue usada por sus ventajas fue el *apprenticing*, la cual pone a uno en los zapatos de los que antes tenían que hacer esa labor. Con este método se pudo comprender el alcance del proyecto como tal, fruto de haber recorrido todo su flujo, generado varias BE y experimentando los distintos casos de uso presentados.

Además de las mencionadas con anterioridad, otro método que fue de gran ayuda para validar la idea del cliente fue el *Dirty Process Modeling* la cual, diagramando sin prestar mucha atención en los detalles, permitió confirmar que tanto el cliente como los alumnos estaban en la misma página.

Cuando conceptualmente estuvieron todas las partes de acuerdo, se pasó a una fase más de construcción - sin perder tiempo en el diseño e implementación - y se recurrió a otra técnica de

elicitación: el prototipado. Usando dibujos y una mínima simulación de la solución, se le dio más acercamiento a lo solicitado, haciendo los ajustes necesarios según lo mostrado, y de esta manera orientar el trabajo en el sentido correcto.

Mirando hacia atrás, si bien no se hizo conscientemente, también se empleó el método de *Document Archeology*, debido a que previamente existía un pseudosistema: el procedimiento manual que justamente se buscaba automatizar. Así, además de contar con los clientes para entrevistarlos o aprender de ellos, se contaba con documentación, lo que fue más beneficioso para los estudiantes que solo escuchar la idea de los docentes.

Parte 2

Si bien usar varias estrategias trae consigo los beneficios de poder obtener una gran cantidad de información sobre el sistema deseado y el contexto en el que se debe desarrollar, se corre el riesgo de que tanta información termine abrumando al equipo. Esto puede llegar a apartar significativamente al mismo del objetivo a cumplir, ya que se podrían generar distintos caminos que no conducen al mismo destino; causando que se pierda tiempo y esfuerzo en un trabajo mal abordado.

A su vez, considerando que los docentes con los que se trabajó son del mismo rubro que los alumnos, no solo se encontró una similitud significativa en los conceptos manejados por ambas partes, sino también en la lógica que se aplicó, siendo estas ventajas no menores frente a la posibilidad de que el cliente y los desarrolladores sean de mundos completamente diferentes, donde la comunicación probablemente sea compleja.

De hecho, además de las diferencias que pudiera haber en la comunicación, pasará muchas veces que un cliente sabe mucho de su negocio, pero no sabe bien qué es lo que necesita. En la vida profesional sucederá que se presentarán clientes con ideas que no representan el problema que se debe solucionar, lo cual puede entorpecer la labor, y justamente, las entrevistas a clientes que no saben qué necesitan, corren el riesgo de volverse difíciles e infructíferas.

Esta carencia de las entrevistas se puede solucionar con un entrevistador hábil con las palabras o que dada su experiencia sepa conducir al cliente o entrevistado, a donde necesita para tener un punto de partida y dónde trabajar.

Otro error parcial y bastante riesgoso es que cuando ya existe una solución a algo y el fin es mejorar lo existente, se tiene la tendencia natural de conservar lo que ya está, pero con modificaciones. Aunque en algunos casos pueda resultar útil, en otras ocasiones es probable que se necesite una restructuración o, dicho de otra manera, el adoptar un enfoque distinto para el problema, sumado a que en sistemas anteriores la compatibilidad con las herramientas de desarrollo es compleja, el peligro de vulnerabilidades que todo el tiempo se descubre, pero que sin actualizaciones no se mitigan, es alta. A lo anterior se le suma el hecho de que se pierden las ventajas tecnológicas alcanzadas luego de la creación de la última versión que existía. Todas esas son algunas de las desventajas que conlleva depender demasiado del *Document Archeology.*

Por otro lado, el modelado rápido o *Dirty Model*, si bien brindaba la oportunidad de entender que era lo que el cliente quería, al no ser algo tangible, sino más ideas y bocetos muy abstractos, así como simples, se corría el riesgo de no llegar al nivel de detalle que le permitiera al cliente ver cómo sería su sistema andando. Esto se logra gracias a la implementación de prototipos, los cuales pueden tener la desventaja de consumir mucho tiempo si no se hace de la forma correcta, pasando a ser una versión, más que una maqueta del sistema.