1. Propuesta de servicios

1.1. Propósito

La presente propuesta técnica tiene como propósito exponer nuestra alternativa de solución al problema de la empresa de TIS para una juez virtual para la Olimpiada Internacional de Informática. Además el documento servirá de guía para el proceso de desarrollo del producto de software y formará parte indivisible del contrato estipulado entre la empresa TIS y el proponente, esperando que cumpla y satisfaga con las exigencias emitidas por parte de la empresa licitante.

1.2. Objetivo

1.2.1. Objetivo general

Desarrollar un juez virtual, que funcione a nivel departamental, que permita a olimpistas enfrentarse en un modo de evaluación, como el que se usa la competencia mundial.

1.2.2. Objetivos específicos

- Desarrollo de componentes de gestión de usuarios.
- Desarrollo de componentes del modulo de evaluador.
- Desarrollo de componentes del gestión de competición.
- Desarrollo de componentes del gestión de problemas
- Realizacion de la gestión de calidad del sistema integrado

1.2.3. Tecnología a utilizar

Para el cumplimiento de los objetivos de esta propuesta y las condiciones establecidas en el pliego de especificaciones de la empresa TIS, se ha considerado desarrollar el sistema del «Juez Virtual»con las siguientes herramientas de software:

Herramienta	**Software**
Servidor web	Apache
Gestor de base de datos	MySQL
Lenguaje de programación	PHP
Framework de apoyo	Symfony
Biblioteca de apoyo	jQuery
Sistema de control de versiones	Git
Entorno integrado de desarrollo	Netbeans
Composición de documentación	LaTeX
Editor de texto plano	vim

Las versiones a ser utilizadas de Apache, MySQL y PHP serán definidas de acuerdo al software instalado en el laboratorio de TIS.

1.3. Tipos de usuarios

1. Olimpista.
2. Administrador.
3. Miembro de comité académico.

1.4. Funcionalidades globales

Gestión de usuarios La gestión de usuarios consta de la creación, modificación y depuración de usuarios, los tipos de usuarios se especifican en 1.3. Cada tipo de usuario tendrá establecido sus privilegios con respecto al Sistema.

Modulo evaluador Consta de un sub-sistema que recibe una solución a un problema y ejecuta los casos de prueba correspondientes al problema, luego devuelve un puntaje de acuerdo a los resultados de la ejecucion de los casos de prueba.

Gestión de competición Permite crear una competición entre olimpistas, se podrá seleccionar el modo de la competición, es decir si la competición va ser verdadera o solo sera una practica. Una vez creada la competición se la podrá dar inicio y también se la podrá finalizar. También se contempla la creación de reportes relacionados a la competición, como ser un ranking de una competición determinada.

Gestión problemas Permite la subida nuevos problemas, la modificación de problemas y dar de baja un problema.

2. Metodología de desarrollo

Nuestro marco de trabajo para la gestión y desarrollo de software estará basada en un proceso iterativo e incremental, utilizando un componente en entornos basados en el desarrollo Ágil de software SCRUM.

El **modelo incremental** se basa en incrementos, cada incremento sigue el modelo cascada y posee características de retro alimentación.

Estos modelos iterativos se basan en dividir el proyecto de desarrollo en varias etapas, llamadas iteraciones. Las iteraciones son cortas (unas cuantas semanas, excepto en proyectos enormes) y su duración es fija (no puede alargarse: si hay retrasos, estos se incluyen en otra iteración).

La idea central es que, en cada una de esas iteraciones, se construye una parte pequeña del sistema (esto se llama a veces "desarrollo incremental"). Para esa parte del sistema, se realiza todo el proceso: análisis, diseño, programación y pruebas. Se acaba la iteración con un ejecutable que incluye todas las partes del sistema construidas hasta el momento. Los aspectos del sistema con más riesgo (por ejemplo, la arquitectura) se construyen en las primeras iteraciones. El esquema de un modelo iterativo se muestra gráficamente en la siguiente figura. En este sentido podemos considerar SCRUM como el paradigma de metodología de desarrollo ágil, definiendo la forma de abordar un proceso de desarrollo de software de forma ágil y liviana, a través de la descripción de un conjunto de roles, componentes y organización de la actividad diaria.

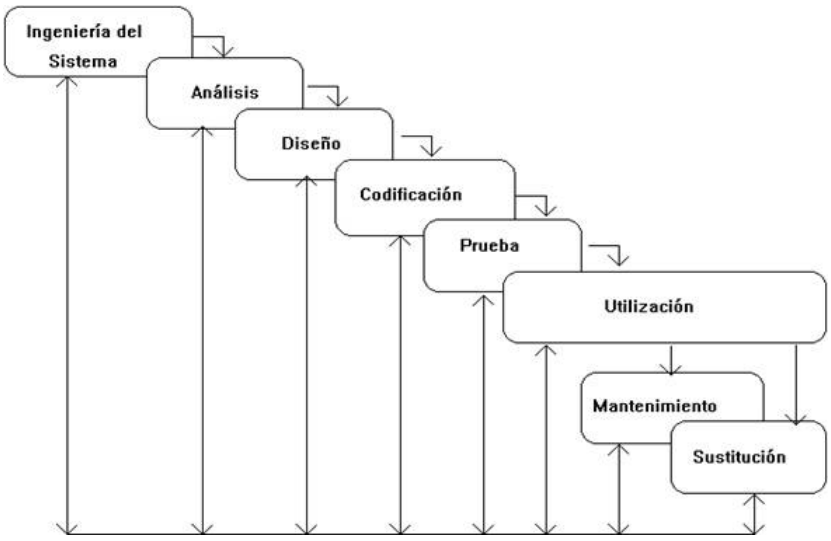

Figura 2.1

2.1. Roles en la metodólogía SCRUM

- **Rol product owner**
  Persona que trabaja junto con el **Cliente** con conocimientos de negocio e informática sus responsabilidades son :
  *Representante de todas las personas interesadas en los resultados del proyecto.
  *Es quien coordina las actividades administrativas o funcionales que deben realizar los usuarios en complementación a las actividades y requerimientos del equipo de desarrollo.
  *Es quien se encarga del financiamiento del proyecto y de que los requerimientos o necesidades de los usuarios sean satisfechos en tiempo y alcance.

  Sobre el Product Owner recae la responsabilidad de definir el conjunto de requerimiento.

- **Cliente**
  Ser el**representante de todas las personas interesadas** en los resultados del proyecto (internas o externas a la organización, promotores del proyecto y usuarios finales [idealmente también debería ser un usuario clave] o consumidores finales del producto).
  Definir los **objetivos del producto o proyecto**.
  Colaborar con el equipo para planificar, revisar y dar detalle a los objetivos de cada iteración

- **Scrum Master**
  Responsable de que el equipo desempeñe sus actividades de una manera eficiente y organizada, también elimina obstáculos que impiden que el equipo alcance el objetivo del sprint.

- **Scrum Team**
  Es un equipo formado de (pm)7 - 3 integrantes que se encargan de revisar los requerimientos, la tecnología disponible y evalúan los conocimientos para determinar de manera colectiva cómo incrementar la funcionalidad del software, son responsables de elegir las historias de usuarios para una determinada iteración y se caracteriza por ser auto gestionado y auto organizado.

2.2. Componentes de scrum

Término general para cualquier tipo de información creada, producida, cambiada o utilizada por el equipo Scrum.

- **Sprint o Iteracón**
  Es la base de desarrollo de Scrum, su duración se encuentra entre 15 a 30 días, solo el Scrum Master puede abortar el sprint si lo considera no viable por alguna de estas razones:
  * La tecnología establecida no funciona
  * El equipo ha tenido interferencias

- **La duración del sprint**
  Los sprint cortos permite al equipo ser ?ágil? es decir cambiar de dirección frecuentemente, pero los sprint largos tampoco están mal el equipo consigue más tiempo para realizar sus actividades y recuperarse de los problemas que surjan en el sprint y así cumplir la meta.

- **La meta del sprint**
  La meta del sprint debería responder a la pregunta fundamental ¿Por qué realizamos este sprint? Las respuestas pueden ser «**Cumplir con las espectativas del cliente.**»

- **Product Backlog**
  Contiene los objetivos/requisitos de alto nivel del producto o proyecto, que se suelen expresar en forma de historias de usuario. Para cada objetivo/requisito se indica el valor que aporta al cliente y el coste estimado de completarlo. La lista está priorizada balanceando el valor que cada requisito aporta al negocio frente al coste estimado que tiene su desarrollo, es decir, basándose en el Retorno de la Inversión (ROI).
  En la lista se indican las posibles iteraciones y las entregas (releases) esperadas por el cliente (los puntos en los cuales desea que se le entreguen los objetivos/requisitos completados hasta ese momento), en función de la velocidad de desarrollo del (los) equipo(s) que trabajará(n) en el proyecto. Es conveniente que el contenido de cada iteración tenga una coherencia, de manera que se reduzca el esfuerzo de completar todos sus objetivos.
  La lista también tiene que considerar los riesgos del proyecto e incluir los requisitos o tareas necesarios para mitigarlos.
  Antes de iniciar la primera iteración, el cliente debe tener definida la meta del producto o proyecto y la lista de requisitos creada. No es necesario que la lista sea completa ni que todos los requisitos estén detallados al mismo nivel. Basta con que estén identificados y con suficiente detalle los requisitos más prioritarios con los que el equipo empezará a trabajar. Los requisitos de iteraciones futuras pueden ser mucho más amplios y generales. La incertidumbre y complejidad propia de un proyecto hacen conveniente no detallar todos los requisitos hasta que su desarrollo esté próximo. De esta manera, el esfuerzo de recoger, detallar y desarrollar el resto de requisitos (menos prioritarios) está repartido en el período de ejecución del proyecto.
  En el caso del desarrollo de un producto, la lista va evolucionando durante toda la vida del

producto (los requisitos .emergen"). En el caso de un proyecto, conforme éste avance irán apareciendo los requisitos menos prioritarios que falten. Esto produce varias ventajas:
Se evita caer en parálisis de análisis al inicio del proyecto, de manera que se puede iniciar antes el desarrollo y el cliente puede empezar a obtener resultados útiles. Se evita analizar en detalle requisitos no prioritarios que podrían cambiar durante el transcurso del proyecto, dado que el cliente conocerá mejor cuál ha de ser el resultado a conseguir, o bien por que podrían ser reemplazados por otros. Puede llegar a un punto del proyecto en que no valga la pena analizar ni desarrollar los requisitos restantes, dado el poco retorno de inversión (ROI) que tienen.

Definición de completado (definition of done)
El cliente y el equipo tienen que acordar la "definición de completado? de los objetivos / requisitos en el proyecto:
Debe asegurar que el incremento de producto es potencialmente entregable al cliente al finalizar cada iteración, que no hay tareas pendientes que puedan impedir utilizar los resultados del proyecto lo antes posible. De este modo, el cliente podrá tomar decisiones correctas cuando al final de cada iteración el equipo le haga una demostración de los requisitos completados: cambiar las prioridades en función de la velocidad de desarrollo, solicitar una entrega del producto desarrollado hasta ese momento, etc.
Debe incluir lo necesario para considerar de manera clara que el cliente obtendrá lo que necesita según sus criterios de entregables y de calidad (producto construido, probado, documentado, refactorizado para conseguir calidad interna/mantenibilidad, etc.).
Además de esta definición de completado, a cada objetivo hay que asociarle sus condiciones de satisfacción, preferiblemente en forma de casos de prueba de aceptación, en el momento de crear la lista de requisitos priorizada (Product Backlog).
Notar que la definición de completado también sirve como base para identificar las tareas necesarias para conseguir cada objetivo/requisito, en la reunión de planificación de la iteración (Sprint planning). Para cada objetivo se crearán más tareas que la definición de completado (o menos) y con más significado. Por ejemplo, respecto a que el objetivo tiene que estar ?construido?, pueden aparecer varias tareas, del estilo ?construir el componente X?, ?modificar la pantalla Y?, ?modificar la BBDD?, ?preparar el script de carga?, etc.

Iteración de entrega (release sprint)
Cuando el cliente solicita una entrega de los objetivos/requisitos completados hasta ese momento, el equipo puede necesitar añadir una iteración de entrega, más corta que las iteraciones habituales, donde realizar alguna tarea que no ha sido necesaria o posible hasta el momento de la entrega final y acabar de corregir defectos detectados en la última demostración.

Uso de la lista

Para medir la velocidad de desarrollo del equipo, ver una progresión constante y extrapolar la fecha de las entregas, es conveniente seguir las siguientes recomendaciones: Los requisitos deben tener un esfuerzo semejante para ser completados.
La estimación de un requisito no debe ser superior a 10 días (si las iteraciones son de 20 días laborables).
Cada requisito tiene asociado un factor de complejidad, que permite ajustar su coste estimado

en función de la incertidumbre de la complejidad de su desarrollo en el momento de introducirlo en la lista. Este factor de coste se irá ajustando conforme las iteraciones avancen y el equipo conozca mejor el producto o proyecto, su contexto y su velocidad de desarrollo con las herramientas y tecnologías que utiliza.
Si un requisito depende de otro, se coloca en algún punto por debajo del que depende.
Si un requisito no se finaliza en una iteración, se le vuelve a poner en alguna de las siguientes iteraciones, indicando el coste pendiente de desarrollo.
El .ºrigen"permite saber quién podría participar en la definición de un objetivo/requisito y sería conveniente que estuviese presente en el momento de su demostración.

- **Sprint Backlog**
  Especifica la serie de tareas que se van a desarrollar, que descompone las funcionalidades del Product backlog en las tareas necesarias para construir un incremento o una parte completa y operativa del proyecto.

2.3. Reuniones scrum

- **Planificación del Sprint**
  Para la planificación se deberán tomar como base las prioridades y necesidades del cliente, se determina cuáles y cómo van a ser las funcionalidades que incorporara el producto tras el siguiente sprint, consta de dos partes:
  *En la primera parte se decide qué elementos de la pila de producto se van a desarrollar. *En la segunda parte se desglosa estos para determinar las tareas necesarias y estimar el esfuerzo para cada una, también procedemos a asignarlas a cada uno de los miembros del equipo.

- **Seguimiento del Sprint o reunión diaria**
  Es una reunión breve no más de 15 minutos en la que cada miembro del equipo dice las tareas en las que está trabajando, si se ha encontrado o prevé encontrarse con algún impedimento o actualizar sobre la pila de sprint las ya terminadas.

- **Revisión del Sprint**
  Reunión que se realiza al final del sprint en la que el equipo presenta al propietario del producto.

- **Retrospectiva**
  Es una reunión en la que debemos analizar las cosas que hemos hecho bien y las cosas que hemos hecho mal durante el sprint.

2.4. Ventajas de scrum

Simple: Se centra especialmente en facilitar el desarrollo rápido, por lo que su complejidad (por ejemplo desde el punto de vista de la documentación a generar o de la organización de equipos) se ha tratado de reducir al máximo.

Ágil: La división del trabajo en pequeñas unidades funcionales (sprints) permite mantener una política de entregas frecuentes de software que ofrecen una visión clara del estado del proceso y permite la introducción de modificaciones.

Flexible: Todo el desarrollo se contempla como un ciclo de iteraciones continuas de desarrollo, lo que facilita la introducción de modificaciones ?sobre la marcha?, mejorando continuamente el proceso.

Colaborativa: El planteamiento, desde el punto de vista de la organización del equipo, resulta bastante horizontal (en contraposición a una organización jerárquica férrea), otorgando a los miembros del equipo de desarrollo una elevado grado de autonomía y auto-organización de su trabajo.

3. Plazo de conclusión del contrato

Una vez entregado el software y firmado un documento de conformidad por parte de la empresa TIS, se dará por concluida la relación contractual con la empresa TIS. Tales acciones en ningún caso podrán exceder el día 6 de diciembre del 2013.

4. Propuesta económica

4.1. Costos operativos

Estos costos se refieren a los servicios básicos y otros suministros, un calculo aproximado en bolivianos por mes sería:

Luz	300
Agua	150
Internet	1000
Teléfono	100
Suministros de oficina	100
Agua embotellada	40
Total	1690

4.2. Costos del personal

Estos costos se refieren al salario del personal de la empresa, un calculo aproximado en bolivianos por mes sería:

Gerente general	6960
Secretaria	2784
Limpieza	1044
Administración	4176
Contabilidad	3480
Ingenieros (3)	14616
Total	33060

4.3. Costo total

Nuestro grupo-empresa realizara el desarrollo del sistema por 20,45 horas a la semana según nuestro calendario definido en la parte A. También se toma en cuenta que el proyecto esta programado para realizarse durante 17 semanas, incluyendo la realización de la propuesta.

Horas totales a trabajarse en el proyecto:

$$20{,}45 * 17 = 347{,}65 \text{ Horas}$$

Suma de costos del personal y los costos operativos:

$$33060 + 1690 = 34750 \text{ Bs.}$$

En el siguiente calculo se detalla el costo por hora:

$$((34750)/22 \text{ horas laborales })/8 \text{ horas al día} = 197{,}44 \text{ Bs./Hora}$$

Calculo del costo total:

$$197{,}44 \text{ Bs./Hora} * 347{,}65 \text{ Horas} = 68641{,}12 \text{ Bs.}$$

Costo	**Importe (Bs.)**
Costo total	68641,12
Precio de venta incluyendo utilidades (20 %)	13728,22
Precio final	**82369,34**

El precio final sería de: Ochenta y Dos Mil Trescientos Sesenta y Nueve 34/100 Bs

5. Plan de pagos

Los pagos se realizarán en la finalización de cada hito.

Pago	**Evento**	**Monto (Bs.)**
1	1.er hito, 9 de septiembre	8075,3
2	2.er hito, 6 de diciembre	52489,42
3	3.er hito, 20 de diciembre	8076,73