**A N E P Escuela Superior de Informática**

CONSEJO DE EDUCACION Av. Gral. Rivera 3729 bis. TECNICO PROFESIONAL Montevideo, Uruguay

**(Universidad del Trabajo Del Uruguay)**

**ByteSoft**

***Fundamentación del modelo de desarrollo***

**Introducción**

En este documento se fundamenta el modelo de desarrollo a seguir durante la totalidad del proyecto. En el mismo se hará mención al funcionamiento del modelo elegido, así como sus características y ventajas con respecto a otros modelos de desarrollo.

**¿Qué son las metodologías de desarrollo ágiles?**

**Origen**

Surgen como contraparte de los modelos tradicionales en la época de los 90, los cuales eran muy estructurados, estrictos , pesados y orientados a los procesos basados en el modelo de desarrollo en cascada que en muchos casos resultaba en grandes desviaciones en tiempo y costes o incluso con un producto final que no cumple con las necesidades del cliente, ya sea por un mal análisis inicial, una mala planificación o por el simple hecho de que las necesidades pueden variar durante el tiempo en que el proyecto se está desarrollando. En el año 2001, miembros destacados de la comunidad se reunieron y adoptaron el nombre de métodos ágiles. Muchos métodos similares a los ágiles fueron creados antes de esta reunión, como el Scrum, XP, Crystal Clear.

**Definición**

Son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

**Ventajas**

Mejoran la satisfacción del cliente porque se lo involucraría lo largo del proyecto, se lo informará de los logros y progresos en cada etapa.

Otra de las ventajas es la mejora de la motivación e implicación del equipo de desarrollo, ya que permite a todos los miembros del equipo conocer el estado del proyecto en cualquier momento.

Estos métodos permiten ahorrar tiempo y costes, se trabaja de forma más eficiente y rápida y con ello de cumple de forma estricta el presupuesto y los plazos pactados dentro de un proyecto.

Se trabaja con velocidad y eficiencia mediante entregas parciales del producto, de este modo es posible entregar en el menor intervalo de tiempo posible una versión mucho más funcional del producto.

El cliente juzgará las entregas parciales y podrá decidir si eliminar o modificar cualquier característica del producto. Esta continua interacción entre los desarrolladores y los clientes asegura que el producto final cumpla con los requisitos propuestos por el cliente.

**¿Por qué un modelo de desarrollo ágil y no uno tradicional o ambos?**

Primero que nada, al analizar los métodos de desarrollo ágiles y tradicionales, se puede llegar a una simple conclusión y es que son opuestos el uno del otro.

Tampoco hay posibilidad alguna de combinar estos métodos ya que, si bien podría funcionar, el hecho de combinar una metodología tradicional (que se caracteriza por ser rígida e inflexible) estaría quitando lo “ágil” de un método de desarrollo ágil, por lo cual descartamos esa posibilidad.

Tenemos en cuenta que la letra nos exige el uso de algunos softwares como Trello y GitHub, lo cual nos inclina a una metodología ágil.

Un punto a favor de los modelos ágiles es que actualmente hay una tendencia al uso frecuente de modelos ágiles y como nuestro proyecto emula una empresa verdadera trabajando en la realidad impuesta termina favoreciendo la elección de un modelo de desarrollo ágil.

También tuvimos en cuenta el propósito de la institución a la que asistimos, de prepararnos para un mercado laboral tecnológico el cual está en constante cambio y al que deberemos adaptarnos.

**Fundamentación general**

Hemos llegado a la conclusión de que ningún modelo encaja totalmente con nuestra realidad, y como estamos en la necesidad de usar un modelo de desarrollo, como lo requiere la letra del proyecto, mediante una ardua discusión nos vimos con algunos grandes factores a tener en cuenta a la hora de decidir nuestro modelo de desarrollo:

1. Tenemos entregas con Productos Mínimos Viables (PVM), además del requerimiento de una pizarra colaborativa, hecha en Trello, lo cual se adecúa más a las metodologías de un modelo de desarrollo ágil y, en este caso, más específicamente Kanban (el cual será explicado más adelante).
2. La letra nos brinda unos requerimientos específicos con sus respectivas fechas de entrega ya definidas, lo cual nos inclina más a un modelo de desarrollo tradicional.
3. No tenemos un equipo de desarrollo lo suficientemente capaz y experto como para manejar un proyecto a gran escala (como es normal en un modelo de desarrollo tradicional) pero tampoco tenemos una figura de Project Manager o Product Owner como para que nos guíe en el transcurso del proyecto (como es normal en la mayoría de modelos de desarrollo ágiles). En resumidas cuentas, no contamos con el personal suficiente para seguir la mayoría de modelos ágiles.

Por todos estos motivos, no hay un claro modelo a seguir. Luego de varias discusiones, llegamos a la conclusión de implementar nuestro propio modelo de desarrollo basándonos en otra metodología de desarrollo ágil, Scrumban (Luego se desarrollará el mismo, es un modelo híbrido de Scrum y Kanban), personalizándolo y adecuándose a nuestra realidad y necesidades.

**¿Qué es Scrum?**

Scrum es un framework (marco de trabajo) ágil para desarrollar, entregar y mantener productos complejos.

Scrum se define en roles, eventos y artefactos propios de él. Estos componentes sirven a un propósito específico y es esencial para el éxito de Scrum.

**Scrum Team**

Es el equipo que consiste en un Dueño de Producto, Equipo de Desarrollo y un Scrum Master.

**Roles de Scrum**

1. **Product Owner (dueño del producto)**

Es la persona que debe maximizar el valor del producto hecho por el equipo de desarrollo.

Además, es quién se encarga de otorgarle requerimientos o expresar sus necesidades al equipo de desarrollo. Es quien define qué va a tener el producto, qué no va a tener, sus características, qué agregar o sacar para futuros sprints, etc. El producto se basa en sus necesidades, es quien maneja el producto y tiene un gran control sobre el mismo.

1. **Development Team (equipo de desarrollo)**

El equipo de desarrollo decide cómo convertir el backlog en un incremento de producto.

Los miembros individuales pueden tener más o menos conocimientos en un área, pero la responsabilidad recae en el equipo como un todo.

Son los encargados de realizar las tareas priorizadaspor el **Product Owner**. Es un equipo multifuncional y auto-organizado.

1. **Scrum Master**

Ayuda a los integrantes a entender la teoría, prácticas, reglas y valores de Scrum. También ayuda a externos al equipo a entender las interacciones con el equipo de Scrum, cuales pueden ser útiles y cuáles no.

**Sprint**

Es un bloque de tiempo de un mes o menos durante el cual se creará un incremento de producto “Terminado” utilizable. Al finalizar un sprint empieza otro.

Durante el sprint:

1. No se pueden hacer cambios que afecten a el objetivo del sprint (Sprint Goal).
2. No se pueden disminuir objetivos de calidad.
3. No se puede cambiar su duración.

**Eventos de Scrum**

1. **Planificación del sprint**

Se lleva por todo el equipo y responde a las siguientes preguntas:

* ¿Qué puede entregarse en el incremento resultante del sprint que comienza?
* ¿Qué puede hacerse en este incremento?
* ¿Cómo se conseguirá hacer el trabajo necesario para entregar el incremento?

1. **Daily Meeting**

Son reuniones diarias de 15 minutos del equipo de desarrollo. En este tiempo se planea el trabajo hasta la próxima reunión de equipo y se realiza el seguimiento del progreso del Sprint. En este lapso de 15 minutos los integrantes del equipo deben responden las siguientes preguntas:

* ¿Qué se hizo anteriormente que ayudó al equipo de desarrollo a lograr el Objetivo del Sprint?
* ¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el Objetivo del Sprint?
* ¿Hay impedimentos que evite que el equipo de desarrollo cumpla los objetivos?

Usualmente los integrantes del equipo de desarrollo se vuelven a reunir después del scrum diario, para tener discusiones detalladas, adaptar el resto del trabajo del sprint.

El scrum master es el que enseña al resto del equipo los límites del bloque de tiempo de 15 minutos. Si personas ajenas al equipo están presentes, el scrum master se asegura de que no interrumpa en la reunión. Los scrums diarios mejoran la comunicación, disminuyen la necesidad de realizar otras reuniones, identifican impedimentos en el desarrollo; promueven la rápida toma de decisiones y mejorar el conocimiento del equipo.

1. **Sprint Review**

Es para inspeccionar el incremento y adaptar el Product Backlog si fuese necesario. En esta revisión el Equipo Scrum y los interesados colaboran acerca de lo que se hizo durante el Sprint. Basándose en el Product Backlog se hacen las modificaciones. Es una reunión informal, no es una reunión de seguimiento, y la presentación del incremento sirve para fomentar la retroalimentación y la colaboración.

Son reuniones de máximo 4 horas en scrums de un mes. El tiempo se adapta a la duración del scrum. El Scrum Master hace que el evento se lleve a cabo y los asistentes entiendan su propósito, y que se respeten en los tiempos fijados para el bloque.

La revisión incluye los siguientes elementos:

* El dueño del producto
* Explica los elementos terminados y los que no se terminaron.
* El equipo de desarrollo
* Habla de lo que estuvo bien en ese Sprint, que problemas aparecieron y su resolución.
* Hace la demostración del trabajo finalizado y responde preguntas.
* Revisión de la evolución del proyecto a lo largo del tiempo, de las capacidades potenciales y del mercado para las próximas entregas de funcionalidad prevista del producto.

En resumen, el Sprint review se centra en el producto y en maximizar el valor de los resultados del trabajo en el Sprint.

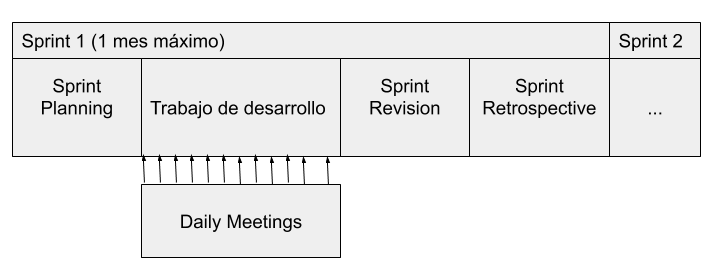
1. **Sprint Retrospective**

Es el último evento que tiene un sprint, integrado por todo el Scrum Team. En este se hace una puesta a punto de las tareas planteadas en todo el sprint anterior y se inspeccionan las mismas, para así poder mejorar el desarrollo e iniciar un nuevo sprint evitando problemas anteriores. Resumidamente, se hacen 3 preguntas, “¿Qué salió bien?”, “¿Qué salió mal?” y “¿Qué se puede mejorar?”.

El Scrum Master alienta al equipo para que mejore, dentro del marco de proceso Scrum, su proceso de desarrollo y sus prácticas para hacerlos más efectivos para el siguiente Sprint

Se busca inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos, herramientas y en la implementación del propio Scrum. También en Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras.

En resumen, el Sprint Retrospective está centrado en el proceso y la continua mejora de este.

**Objetivo del sprint (Sprint Goal)**

Es la meta del sprint que debe ser alcanzable mediante la realización de un conjunto “coherente” de ítems del Product Backlog.

Es una guía para el equipo de desarrollo del porqué se está construyendo el incremento. El objetivo del sprint brinda al equipo de desarrollo flexibilidad con respecto a la funcionalidad implementada en el Sprint. Se pueden negociar el alcance del Sprint Backlog durante el sprint.

**Artefactos de Scrum**

* **Lista de Producto (Product Backlog)**

En esta lista estarán todos los elementos que se conozcan que sean necesarias para el producto. El **PO** es el responsable de esta lista y la ordenará en base al valor, riesgos, dependencias y necesidades del negocio.

* **Lista de Pendientes del Sprint (Sprint Backlog)**

En esta lista estarán todos los elementos del Product Backlog que fueron seleccionados para el Sprint. La lista de Pendientes es una predicción hecha por el Development Team acerca de qué funcionalidad tendrá que cumplir el producto para concretar un **incremento**.

* **Incremento**

Es la suma de los elementos de la lista de producto durante el sprint. El incremento debe estar terminado según la definición de terminado del equipo, el incremento debe ser utilizable sin importar si se va a salir al mercado o no.

**¿Por qué no usar únicamente Scrum?**

Tenemos 2 puntos por los que no optamos por utilizar solamente el modelo Scrum:

* La letra del proyecto nos exige tener una pizarra colaborativa usando Trello para la organización del trabajo, hecha con notas y columnas, en resumen, una tabla Kanban.
* Es imposible realizar un Scrum, ya que solo se considera Scrum como tal cuando están todas las normas se cumplen al pie de la letra, lo cual no pasa en nuestro caso principalmente porque no tenemos los recursos humanos suficientes como para poder cumplir todos los roles requeridos.

**¿Qué es Kanban?**

La estrategia Kanban también es una metodología ágil para el desarrollo de software basado en tablas y que cada columna está destinada a una función, su versión más simple tiene 3 columnas: TO-DO (Para hacer), IN PROGRESS (En proceso) y DONE (Terminadas).

Cada actividad se clasifica dentro de una de estas columnas y se ubica en un tablero general llamado tablero Kanban. La tabla es flexible en el número de columnas.

Se enfoca en mejorar la visibilidad del flujo de trabajo, limitar el trabajo en curso de acuerdo a la capacidad disponible y medir el tiempo de ciclo de vida de una actividad.

Con todo esto explicado previamente, procederemos a explicar con detalle qué es Scrumban y cómo nos sirvió de base e inspiración para nuestro propio modelo.

En ByteSoft pensamos, luego de analizar estos 2 modelos, sus características, ventajas y desventajas, decidimos no aplicarlos estrictamente, sino, modificarlos y personalizarlos en base a nuestras necesidades, entorno de trabajo, preferencias, etc.

**¿Cuál es nuestro modelo (ByteScrum) y en qué consiste?**

ByteScrum es nuestro modelo personalizado en base a nuestras necesidades partiendo de un híbrido entre Scrum (Marco de trabajo) y Kanban (Metodología de trabajo).

Combinamos parte de la estructura de trabajo de Scrum con el flujo de trabajo visual que nos ofrece Kanban con su tabla.

**¿Cómo funcionará?**

Se utilizará la pizarra colaborativa Trello para tener un control de las actividades a realizar durante cada entrega y cada sprint. En principio, la pizarra contará con 5 columnas, las cuales son:

1. **PRODUCT BACKLOG** (Concepto proveniente de Scrum).  Esta lista tendrá las tareas a realizar durante la entrega actual.
2. **SPRINT BACKLOG** (Concepto proveniente de Scrum). Aquí se almacenarán la totalidad de tareas pertenecientes al sprint actual a realizar por uno o más integrantes, las mismas contarán con una fecha límite de acuerdo a lo establecido en la tabla de actividades.

* El Sprint Backlog “reemplazará” al **TO-DO** procedente de Kanban.
* Tanto en el Product Backlog como en el Sprint Backlog todos los elementos se ordenarán por prioridad.

1. **DOING:** Será la lista que almacene las tareas que están en ejecución, con su respectivo tiempo límite e integrantes relacionados.

* No estableceremos un límite de tareas asignadas a una persona misma persona en esta columna (no utilizaremos el concepto de límite de trabajo en progreso (Work in Progress) de la metodología Kanban).

1. **VERIFY:** En esta lista se guardarán las tareas casi finalizadas que necesiten la aprobación del resto de integrantes para darse por concluidas, esto para evitar posibles errores que se puedan cometer mientras se realiza la tarea.
2. **DONE:** Será donde se almacenen las tareas ya finalizadas y verificadas.

**Roles de ByteScrum**

Eliminaremos el rol de Scrum Master. Este rol no se podrá cumplir en nuestra realidad, todos los miembros tendrán el mismo conocimiento de Scrum y las dudas recurrentes sobre el framework serán resueltas entre sí o con el docente competente.

* **Product Owner**

En nuestra realidad **relacionaremos** el rol del docente con el de product owner, esto de forma **no oficial** ya que el **PO** es parte del Scrum Team y no comprometeremos al docente a cumplir su papel.

El profesor se encargará de proporcionarnos feedback en cada entrega del producto.

* **Development Team**

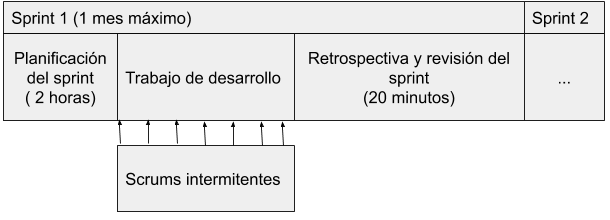
En nuestra realidad estará compuesto por todos los miembros del equipo, nos encargaremos de entregar un incremento del producto terminado.

**Eventos de ByteScrum**

Los sprints contendrán los siguientes eventos:

1. Planificación del sprint (Sprint Planning).
2. Reuniones intermitentes (interm mittent Meeting).
3. Revisión del sprint (Sprint Review) y retrospectiva del sprint (Sprint Retrospective).

* Se sustituirá el Daily Meeting por una reunión intermitente cada 48 horas, la misma durará no más de 15 minutos y puede ser en persona, llamada o mensajes de texto manteniendo la dinámica de Daily Meeting.
* Tanto la Revisión como la Retrospectiva del sprint se unificarán y se realizarán en una misma reunión, manteniendo el orden y la separación marcada de ambos tópicos con una duración máxima de 20 minutos. Además de que serán llevadas a cabo solamente por el equipo de desarrollo.
* El concepto de Sprint seguirá siendo el mismo.



**Comparativa entre ByteScrum y XP**

Hemos decidido utilizar este modelo a diferencia de otras metodologías agiles como podría ser xp esto fue debido a ciertas diferencias entre ellos que nos decantó a utilizar un modelo en vez de otro, si bien nuestro modelo es similar en varios aspectos al xp estos presentan determinadas diferencias que son las siguientes:

Bytescrum una de las principales diferencias es que no se podrán hacer cambios de las actividades planeadas para el Sprint una vez el Sprint Planning haya concluído. a diferencia del xp donde se aceptan cambios con más facilidad en sus iteraciones

En nuestro modelo las reuniones serán cada 48 horas mientras que xp mantiene las reuniones diarias comunes a los modelos ágiles puros.

ByteScrum emplea la tabla Kanban no presente en un modelo XP puro, esto es una ventaja que permite ver el flujo de trabajo a lo largo del proyecto.

Ambas metodologías ágiles comparten varias características, ambas constan de sus respectivos procesos, roles y artefactos, teniendo como objetivo primordial facilitar el proceso de desarrollo de software.