Propuesta de una herramienta basada en la metodología Scrum para la gestión del desarrollo de software

Ing. José Raúl López Morales[1], M.C. José Francisco Gazga Portillo[2],
M.T.I. Juan Miguel Hernández Bravo[3] y M.I.D.S. Alma Delia de Jesús Islao[4]

***Resumen*—En este artículo, se plasma el trabajo interdisciplinario de la maestría en sistemas computacionales con apoyo del Conacyt, impartida en el Instituto Tecnológico de Acapulco. El artículo tiene por meta presentar alternativas de arquitecturas de software para el desarrollo de una aplicación que permita la gestión y monitorización dedicada para proyectos de desarrollo de software basado en la metodología Scrum.**

***Palabras clave*— Arquitectura de software, Metodología ágil Scrum, Modelo-Vista-Controlador (MVC), Desarrollo ágil.**

Introducción

En la década de los 90's surgió el termino **desarrollo ágil** para reaccionar en contra de las metodologías tradicionales, las cuales se consideraban pesadas y rígidas debido a una gran dependencia de planeación detallada antes del desarrollo. Los procesos de desarrollo del software rápido se diseñan para producir rápidamente un software útil. El software no se desarrolla como una sola unidad, sino como una serie de incrementos, cada uno de ellos incluye una nueva funcionalidad del sistema, en los procesos ágiles, por lo general se acepta que una de las primeras etapas en el proceso de desarrollo debe preocuparse por establecer una arquitectura global de un programa (Sommerville, 2016). Existen varias formas de metodologías de desarrollo ágil como el **método de desarrollo de sistemas dinámicos** (en inglés Dynamic System Development Method o DSDM), **Scrum** o **la programación extrema** (en inglés eXtreme Programming o XP) pero todas comparten características similares. La piedra angular de cada rama es la idea de la satisfacción del cliente (Blankenship, Bussa, & Millett, 2011). Actualmente existen diversas herramientas para la gestión y monitorización de proyectos de desarrollo software en internet, las cuales en su mayoría tienen un costo elevado, aunque existen algunas sin ningún costo, pero tienen ciertas restricciones al momento de su uso. **Sprintometer** es una aplicación para la gestión de proyectos Scrum y XP, ofrece una gran variedad de funciones, el único inconveniente es que no permite gestionar un proyecto de manera colaborativa, es decir, si un usuario de la aplicación **Sprintometer** crea un proyecto para gestionar un proyecto, otros usuarios que quieran visualizar y trabajar sobre este proyecto no podrán hacerlo a menos que utilicen el equipo de este usuario. A continuación, se describe **Kunagi**, una herramienta en línea para la gestión de proyectos basados en Scrum, al igual que **sprintometer** sin costo y de la misma manera sólo se puede administrar un solo proyecto a la vez, no es intuitivo en su interfaz, sólo permite usar nombres de equipo ya establecidos, no hay un seguimiento del sprint (bloques o iteraciones) el cual es la parte más importante de la metodología Scrum y por último no proporciona gráficos de los avances del proyecto.

Por otro lado, la herramienta Scrum en línea **icescrum** permite una gestión completa de proyectos Scrum, así como equipos, usuarios y proyectos ilimitados sin embargo para usar la herramienta es necesario registrarse para un periodo de prueba o pagar una suscripción al sitio.

Uno de los factores clave para un buen diseño de una aplicación es la arquitectura de software. La arquitectura de software se interesa por entender cómo debe organizarse un sistema. En el modelo de proceso de desarrollo de software, la arquitectura de software es la primera etapa en el proceso de diseño del software. Es el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema (**clientes y servidores, bases de datos, filtros y nivel jerárquico en el sistema**) y la relación entre ellos. La arquitectura de software es la columna vertebral para construir un sistema de software, la cual es en gran medida la responsable de permitir o no ciertos atributos de calidad del sistema como por ejemplo el **desempeño, modificabilidad, usabilidad y seguridad**. Además, es un modelo abstracto que es reutilizable que puede ser transferido de un sistema a otro.

[1] Ing. José Raúl López Morales es estudiante de Maestría en Sistemas Computacionales en un programa PNPC en el Instituto Tecnológico de Acapulco, jraul_lopz@hotmail.com **(autor corresponsal).**
[2] MC. José Francisco Gazga Portillo es docente de Maestría del Instituto Tecnológico de Acapulco, ita.gazga@gmail.com
[3] MTI. Juan Miguel Hernández Bravo es docente de Maestría del Instituto Tecnológico de Acapulco, jmhernan@yahoo.com
[4] MIDS Alma Delia de Jesús Islao es docente de Maestría del Instituto Tecnológico de Acapulco, alma.islao.ita@gmail.com

Objetivo general

El objetivo principal que se pretende lograr con la presente propuesta consiste en desarrollar una herramienta basada en la metodología Scrum para la gestión de proyectos del desarrollo de software, para ello es importante analizar las arquitecturas de software que pueden ser implementadas para cumplir el objetivo.

Arquitecturas de software

A partir de esta sección se describen algunas de las arquitecturas como alternativa para dar una estructura y a su vez, ayude al desarrollo de la herramienta propuesta.

Arquitectura en capas

Organiza un sistema en capas donde las funcionalidades están relacionadas con cada capa. Una capa proporciona servicios a la capa superior, las capas de nivel más bajo representan funcionalidades o servicios centrales que pueden ser usados en todo el sistema. Las características de esta arquitectura son:

- Describe la descomposición de servicios de forma que solamente entre capas vecinas existe la interacción.
- Las capas de una aplicación pueden estar en una misma máquina o pueden estar distribuidas en diferentes equipos de cómputo.

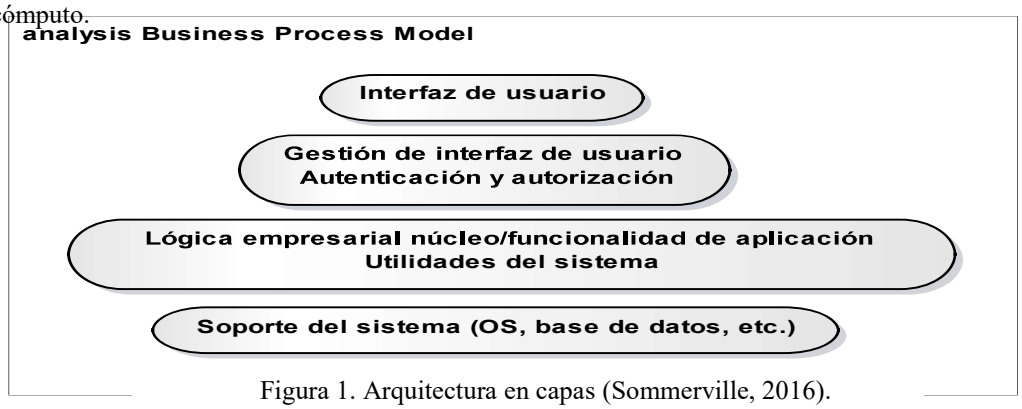

Figura 1. Arquitectura en capas (Sommerville, 2016).

Arquitectura de repositorio

En esta arquitectura los datos de un sistema son gestionados en un repositorio central el cual es accesible por todos los componentes de un sistema. No existe una interacción directa entre componentes, sino a través del repositorio. Esta arquitectura es habitualmente utilizada en **sistemas expertos**, **sistemas multiagente** y generalmente en **sistemas basados en el conocimiento**.

Figura 2. Arquitectura de repositorio para un IDE (Sommerville, 2016).

Arquitectura cliente-servidor

Las tareas se reparten entre los proveedores de servicios, llamados servidores y los demandantes de estos servicios son llamados **clientes**. Un cliente realiza peticiones a otro programa, el **servidor** es quien le da una respuesta. Esta arquitectura puede ser aplicada a programas que se ejecutan en una sola computadora.

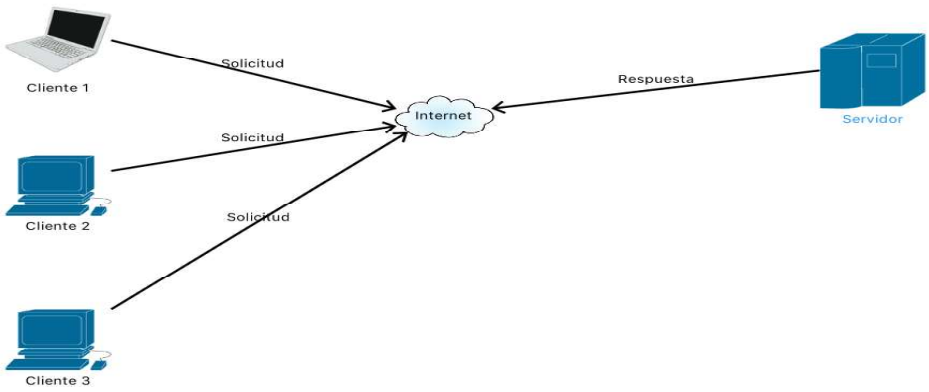

Figura 3. Arquitectura cliente-servidor.

Arquitectura de tubería y filtro

Está diseñada de un conjunto de componentes llamados **filtros** que están conectados entre sí por **tuberías**, que transmiten los datos de un componente a otro. Cada filtro trabaja independientemente de los componentes que están situados antes o después de él. Sus características son:

- Cada paso de un proceso se encapsula en un filtro.
- Los filtros se pueden usar en otros sistemas.
- Los datos se pasan a través de los tubos entre los filtros adyacentes.

Se suele utilizar en aplicaciones de procesamiento de datos (basadas en lotes [batch], así como en transacciones).

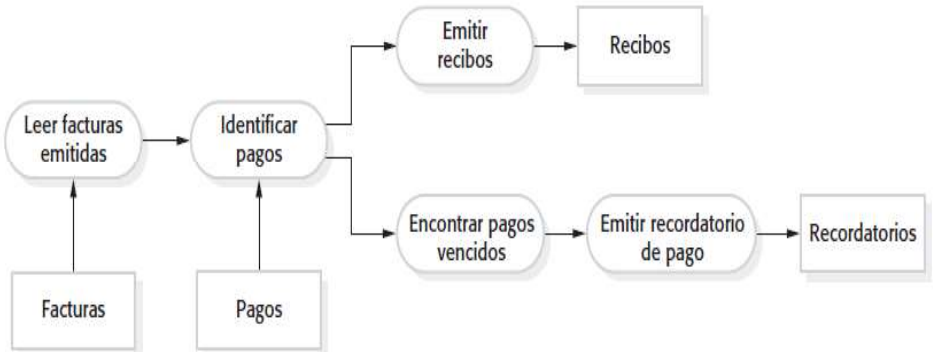

Figura 4. Arquitectura de tubería y filtro (Sommerville, 2016).

Arquitectura de tres capas

Separa una aplicación en tres capas:

- **Presentación**: Su responsabilidad es la representación de datos al usuario mediante los componentes que residen en esta capa y su interacción con esta. Estos componentes permiten al usuario interactuar con los procesos de la capa de negocio de una forma segura e intuitiva.
- **Negocio**: Gestiona la lógica empresarial de la aplicación y puede acceder a los servicios de la tercera capa. En esta capa se produce la mayor parte del trabajo de los procesos de una aplicación, varios componentes de la capa de presentación pueden acceder simultáneamente a los procesos de la capa de negocio y esta debe gestionar sus propias transacciones.
- **Datos**: Esta capa solamente puede comunicarse con la capa de negocio para proveer acceso a los datos y que pueda procesarlos para mostrarlos en la capa de presentación.

A continuación, se muestra en la figura 5 un ejemplo de una arquitectura de tres capas.

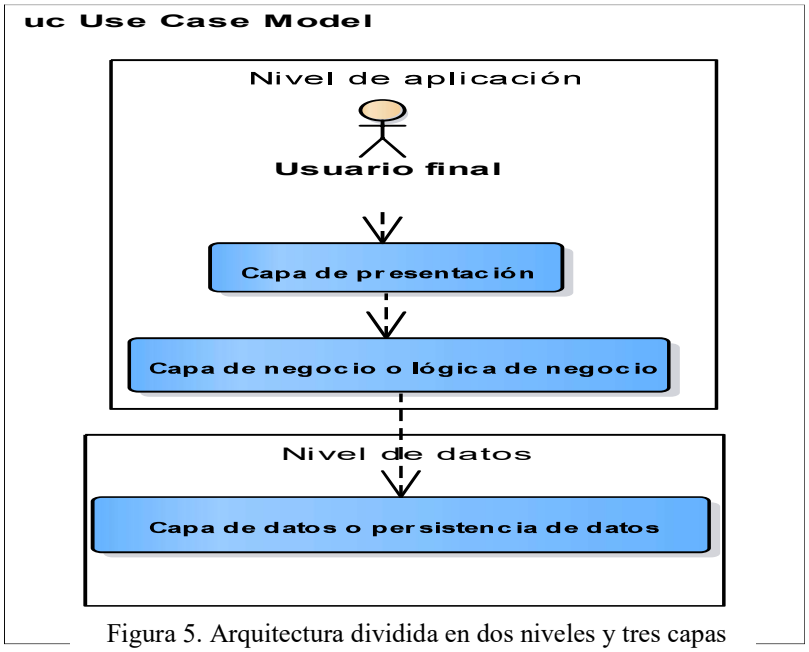

Figura 5. Arquitectura dividida en dos niveles y tres capas (Tahuiton Mora, 2011).

Modelo-Vista-Controlador o MVC (Model-View-Controller)

Es un patrón de diseño arquitectónico que más se utiliza para el desarrollo de **aplicaciones web**. MVC descompone una aplicación en capas permitiendo tener una separación entre la lógica de negocio, la de aplicación y de datos, esta separación de componentes permite tener un código más claro, flexible a cambios y reusable, ver figura 6. El patrón MVC enfatiza tres capas que son importantes en una aplicación las cuales son:

- Modelo: Encapsula o representa los datos de la aplicación el cual es consultado por el **controlador** y a su vez necesitado en la **vista** para ser representados en ella.
- Vista: Se encarga de la interacción con el usuario y la representación de un **modelo** mediante una **interfaz gráfica de usuario**.
- Controlador: Es el intermediario entre el **modelo** y la **vista** ante las peticiones generadas por el cliente mediante la **vista**. Este procesa la solicitud del cliente y se encarga de seleccionar un modelo para poder representar los datos de salida del proceso de la solicitud y estos ser mostrados en la **vista**.

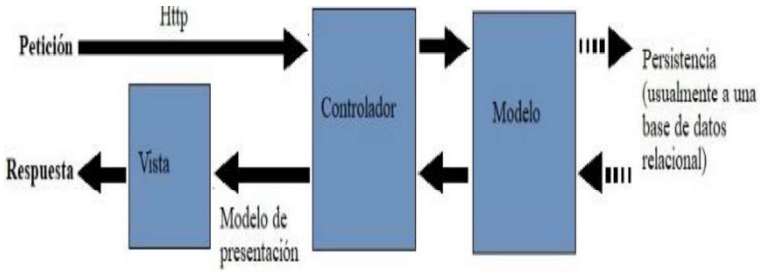

Figura 6. Interacción entre las capas de la arquitectura MVC (Freeman, 2013).

Con las arquitecturas de software presentadas anteriormente, se puede decir que cada una de ellas está enfocada algún tipo de aplicación, por ejemplo, la arquitectura cliente-servidor está enfocada hacia aplicaciones que siempre se instala parte de la aplicación en el cliente, la cual realiza peticiones a la otra parte que es instalada en el servidor, la cual recibe estas peticiones y da respuesta a estas peticiones al cliente. La arquitectura de repositorio está enfocada hacia sistemas expertos o sistemas multiagente. La arquitectura tubería y filtro se suelen usar en aplicaciones de procesamiento de datos. La arquitectura de software seleccionada para el desarrollo de la herramienta propuesta es Modelo-Vista-Controlador (MVC) debido a que es adaptable para el desarrollo de aplicaciones web, por tal motivo fue la arquitectura seleccionada ya que la herramienta que se pretende desarrollar está basada en web; además, la mayoría de los frameworks existentes para la construcción de aplicaciones web de

lado del cliente y de lado del servidor implementan este patrón arquitectónico, separando los intereses o conceptos dentro de una aplicación ayudando a que exista un orden en la codificación, permitiendo que el mantenimiento sea fácil y como gran parte de los patrones de diseño, permite la reutilización de código.

Metodología de desarrollo del software

Para el desarrollo de la herramienta propuesta la metodología a seguir será la metodología Scrum, debido a que la herramienta que se pretende desarrollar gestionará y monitorizará proyectos basados en la metodología Scrum, además está metodología es indicada para proyectos en entornos complejos, la herramienta que se va a desarrollar tiene cierto grado de dificultad y también permite requisitos cambiantes o poco definidos, es decir, es flexible ante cambios y además que las entregas del producto a desarrollar son pequeños incrementos funcionales los cuales se ven reflejados en cortos periodos. A continuación, se describe con un alto nivel de abstracción la metodología Scrum que da soporte a lo mencionado.

Metodología ágil Scrum

Es un marco ágil, donde los programadores se les asigna **iteraciones** o **sprints**, que son periodos de tiempos en los que se enfocan en completar un objetivo predefinido, que posteriormente se convierte en un **trabajo en proceso** (en inglés **Work In Progress** o **WIP**). Uno de los elementos clave de ágil es que estos objetivos casi siempre son características utilizables del producto final (un proceso llamado integración continua). Los programadores trabajan en **equipos Scrum** o **team Scrum** bajo la supervisión del **Scrum master**, que ejecuta reuniones diarias, reuniones cortas para revisar el progreso en el que las personas se paran físicamente en lugar de sentarse (McFedries, 2017).

Se centra principalmente a nivel de las personas y equipo de desarrollo que construyen el producto. Su objetivo es que los miembros del equipo trabajen juntos y de forma eficiente obteniendo productos complejos y sofisticados. Se puede entender Scrum como un tipo de ingeniería social que pretende conseguir la satisfacción de todos los que participan en el desarrollo, fomentando la cooperación a través de la auto-organización. De esta forma se favorece la franqueza entre el equipo y la visibilidad del producto. Los equipos se guían por su conocimiento y experiencias más que por planes de proyecto formalmente definidos. La planeación detallada se realiza sobre cortos periodos permitiendo una constante retroalimentación que proporciona inspecciones simples y un ciclo de vida adaptable (Schwaber & Beedle, 2006).

A continuación, se muestra en la figura 7 el ciclo de vida del desarrollo que propone Scrum para un producto software.

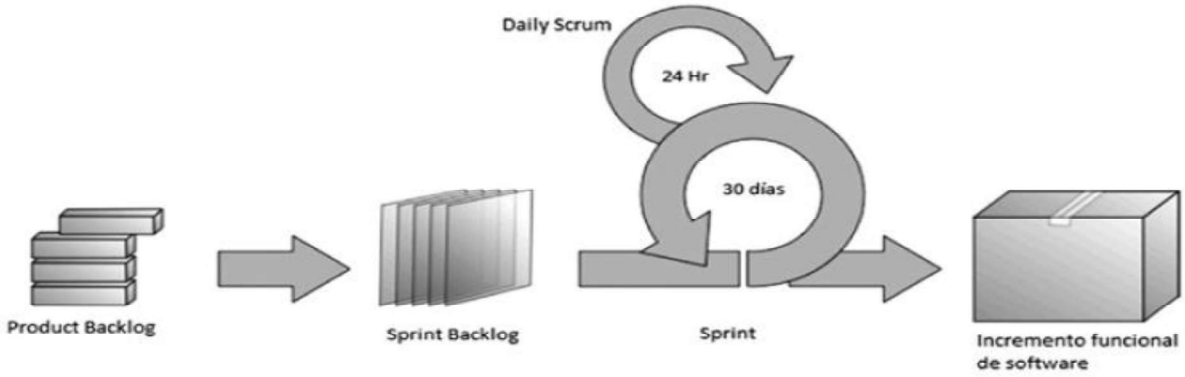

Figura 7. Modelo de desarrollo aplicando Scrum (Navarro Cadavid, Fernández Martínez , & Morales Vélez, 2013).

Metodología de desarrollo del proyecto

- Implementar la metodología Scrum para el desarrollo de la herramienta y obtener el **product backlog** (historias de usuario o requerimientos) y a su vez priorizar las historias de usuario.
- Generar los **sprints backlog** (historias de usuario con mayor prioridad) para generar los incrementos del producto (herramienta Scrum).
- Crear la base de datos con el gestor de base de datos PostgreSQL.
- Crear un proyecto .Net Core MVC en Visual Studio y realizar la conexión con la base de datos.

- Crear los modelos (clases) para manipular la información obtenida por parte de la base de datos
- Implementar la lógica de negocio para satisfacer las historias de usuario en cada sprint utilizando el lenguaje C# de lado del servidor (controlador) o **back-end**.
- Implementar las interfaces gráficas de usuario utilizando las tecnologías que se ejecutan de lado del cliente, para que los usuarios finales puedan manipular y visualizar datos procesados y solicitados al controlador.
- Instalar Windows server en el servidor para poder instalar las aplicaciones necesarias y poder desplegar la aplicación web en el servidor.
- Instalar IIS como servidor web para poder desplegar y ejecutar la aplicación web.
- Vincular el DNS (nombre de dominio) con la IP del servidor y la aplicación pueda ser consulta desde un navegador web utilizando internet.

Comentarios Finales

Conclusiones

En este artículo, se presentó un análisis exploratorio en el que se describieron algunas arquitecturas como alternativas para el desarrollo de la herramienta basada en Scrum para la gestión del desarrollo de software, así como la descripción de la metodología de desarrollo ágil Scrum, debido a que es una metodología adaptable a cambios en los requerimientos ya que en el desarrollo de software los cambios en los requerimientos es algo común. Con los argumentos expuestos y posterior a su análisis, se determinó que la arquitectura de software a implementar en el desarrollo de la aplicación será MVC debido a que la aplicación a desarrollar se trata de una aplicación web, debido a que MVC en comparación a las otras arquitecturas está enfocada en el desarrollo de aplicaciones web, además de que proporciona una mejor plataforma para la implementación que se persigue.

Trabajos a futuro

Este artículo forma parte de una serie de trabajos siendo el primero de estos, donde los trabajos a futuro claramente deben abordar el diseño de la herramienta propuesta aplicando algunos diagramas UML aun cuando la metodología Scrum propuesta a aplicar en la herramienta no exija diagramas UML, estos permitirán obtener una mejor comprensión de las fases o procesos de la metodología Scrum y poder realizar el desarrollo de la herramienta con ayuda del análisis de estos diagramas. Además, se pretende abordar en trabajo a futuro, la conclusión de la herramienta donde se expondrá el desarrollo de la aplicación propuesta, haciendo uso de algunas herramientas de desarrollo de software, implementando la arquitectura propuesta, la metodología de desarrollo de software y los diagramas UML que darán soporte a este.

Referencias

Blankenship, J., Bussa, M., & Millett, S. (2011). *Pro Agile .Net Development with Scrum.* Apress.

Freeman, A. (2013). *Pro ASP.NET MVC 5.* New York: Apress.

McFedries, P. (2017). Agile Development Spaws. *IEEE SPECTRUM*.

Navarro Cadavid, A., Fernández Martínez , J., & Morales Vélez, J. (Septiembre de 2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, 30-39.

Schwaber, K., & Beedle, M. (2006). *Agile Software Development with Scrum.* Conchango.

Sommerville, I. (2016). *Ingenieria de software.* México: Pearson.

Tahuiton Mora, J. (2011). Arquitectura de software para aplicaciones web. Unidad Zacatenco.