Planteamiento de la construcción del modulo

Objetivos: Establecer la metodología a seguir para el desarrollo del módulo de consultas federadas en Apache Marmotta.

Resultados esperados: Documento que explique qué metodología se seguirá para el desarrollo del modulo

El desarrollo del módulo de consultas federadas para datos geoespaciales debe de basarse en alguna metodología que establezca un ciclo de vida del software. Alfonso Fuggetta, basado en el libro *The cathedral and the Bazaar* [1] de Eric Raymon, propone en su *paper: Open source software – an evaluation* [2] que las metodologías de desarrollo de software para tecnologías *open source* deben de ser de rápido prototipado, con desarrollo evolutivo e incremental. Las propuestas de este tipo de metodologías de ciclo de vida de software por Fuggetta son

- Espiral
- Metodologías ágiles

Si bien las metodologías ágiles engloban a diversas metodologías, se discutirán 2 las cuales son las que más se adaptan al proyecto actual.

A continuación, se describirán cada metodología y al final se dará un veredicto acerca de cuál metodología se usará con base al contexto del proyecto.

Método en espiral

Es un método iterativo representado por una espiral el cual permite llevar a cabo lanzamientos del software de manera incremental o refinación progresiva mediante la iteración de ciclos de la espiral. La espiral consta de 4 fases las cuales se deben de cursar para completar el ciclo y volver a empezar [3] (Fig. 1)

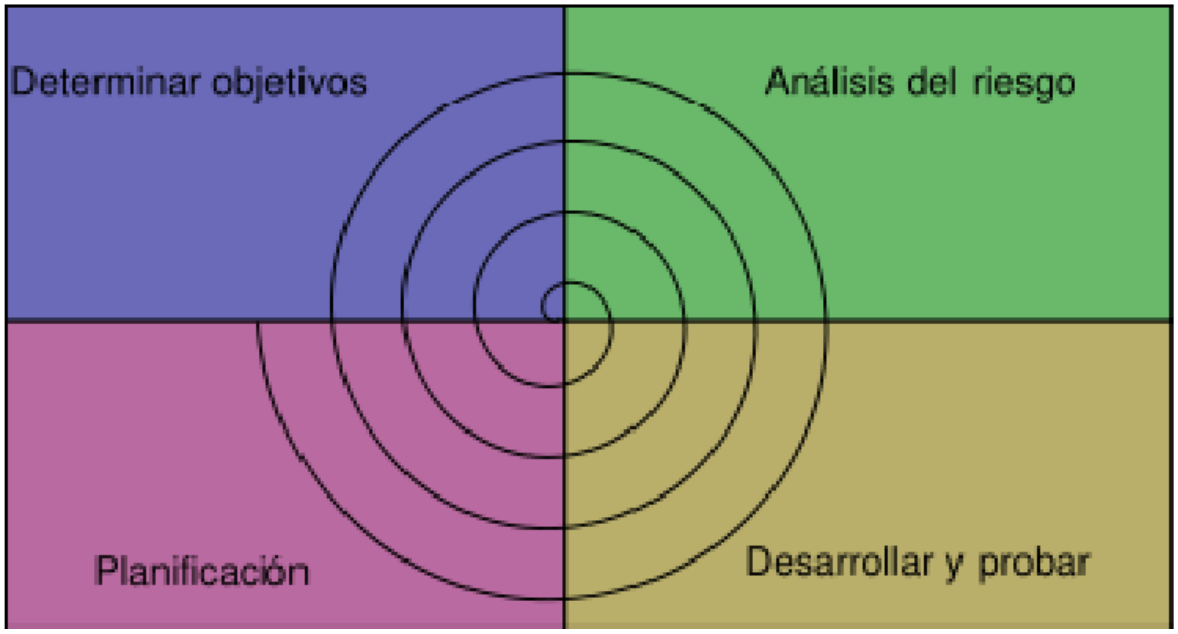

Fig. 1 Diagrama método espiral

Las fases son

- Determinar objetivos:
    - Establecer, en conjunto con el cliente, los objetivos para la fase actual.
    - Determinar restricciones.
    - Identificar riesgos presentes en el proyecto a su vez de contemplar métodos para sortearlos.
    - Acordar qué productos se deben obtener tales como documentación, especificaciones o requerimientos del sistema.
- Análisis de riesgo:
    - Identificar y evaluar riesgos para con el fin de llevar a cabo acciones que disminuyan riesgos significativos. Las acciones pueden ser simulaciones o prototipos del sistema.
    - Analizar riesgos potenciales además de propuesta y selección de alternativas para eliminar o disminuirlos.
- Desarrollo y prueba:
    - Desarrollar el software basándose en los objetivos acordados para la actual fase y los riesgos previamente identificados.
    - Se prueba el software en un entorno de producción para validar su funcionamiento.
- Planificación:
    - Evaluar los productos hechos para organizar como se llevará a cabo la siguiente fase.
    - Detallar qué problemas surgieron en la actual fase para darles solución en la siguiente.
    - Comprobar que los problemas que se presentaron en la anterior fase se solucionaron para corregirlos o considerar si una alternativa es mejor solución al implementarse en la siguiente fase.

Ventajas y desventajas

- Ventajas
    - Los cambios que requiera el cliente pueden ser implementados a temprana etapa del desarrollo del software.
    - Disminución de dificultades al contemplar y tomar acciones contra los riesgos que se presentan.
    - Reutilización de software existente e implementación de funcionalidades de forma progresiva.
    - Se puede incorporar objetivos de calidad de software dentro de su desarrollo.

    - Los requerimientos del sistema pueden ser atendidos de manera precisa.
    - El cliente puede ver resultados del software desde temprana etapa del proyecto.
- Desventajas
    - El tiempo del desarrollo es ambiguo.
    - Si no se realiza un análisis de riesgo adecuado podría afectar negativamente al proyecto por completo.
    - La administración y el proceso del proyecto es más complejo.
    - No es apto para proyectos que involucren riesgos pequeños puesto que podría ser costoso.

Metodologías ágiles

Las metodologías ágiles surgieron cómo una alternativa a las metodologías convencionales tales como el modelo en cascada, el cual todo el sistema tenía que estar desarrollado por completo hasta que se pudieran hacer pruebas [4]. La metodologías ágiles surgen en el 2001 a partir del manifiesto para el desarrollo ágil de software [5] el cual busca

- Mejorar la satisfacción del cliente.
- Adaptar el proyecto a las condiciones de trabajo.
- Proporcionar flexibilidad y rapidez al desarrollo del proyecto.
- Involucrar a los miembros del proyecto a participar en negociación de tareas para mantenerlos motivados.
- Disminuir o eliminar características innecesarias del proyect, puesto a que estas metodologías se basan en entregas parciales.

Si bien existen varias metodologías como *Kanban, Agile Inception, Design Sprint*, *SCRUM* y *Extreme programming*, solo se analizará la última ya que es una de las recomendadas en [2] para el desarrollo de software *open source*.

Extreme Programming (XP)

Metodología surgida en 1996 por parte de Ward Cunningham y Kent Beck la cual consiste en un conjunto de prácticas para atender problemas de entrega de software de calidad velozmente cuyas necesidades de negocio suelen cambiar proporcionando flexibilidad, eficacia y control. La palabra *extreme* surge de seguir las buenas prácticas de ingeniería de software al extremo [6].

XP está pensada para proyectos pequeños los cuales involucren de 2 a 12 personas. Está hecha para ser utilizada en los siguientes tipos de proyectos:

- Requisitos vagos y/o que cambian con frecuencia
- Sistemas con paradigma orientado a objetos.
- Equipos con pocas personas
- Desarrollo del sistema de forma incremental.

Cabe decir que esta metodología está orientada a base de principios y prácticas los cuales son:

- Comunicación entre miembros del equipo.
- Retroalimentación por parte del cliente a los miembros del equipo.
- Simplicidad.
- Coraje.
- Respeto.

Los roles presentes en esta metodología son:

- Clientes
    - Fijan prioridades y marcan necesidades del proyecto.
    - Regularmente son los usuarios finales del sistema.
- Programadores:
    - Son quienes llevan a cabo la *XP.*
    - Llevan a cabo pruebas unitarias.
- *Testers:*
    - Llevan a cabo pruebas funcionales.
    - Registran resultados.
    - Comunican resultados.
    - Responsables de herramientas de pruebas.
- *Coach:*
    - Ayudan a los demás integrantes del equipo.
    - Establece la ruta del proyecto.
- Manager:
    - Ofrece recursos.
    - Lleva a cabo comunicación con miembros externos del equipo.
    - Coordina el equipo.

Las prácticas que se llevan a cabo en *XP* son:

- Planeación
    - Se recaban los requerimientos del sistema para que los programadores del *XP* comprendan las principales características, funcionalidad y el contexto de negocio en el que se desarrolla el proyecto.
    - Se llevan a cabo “historias” por parte del cliente, el cual especifica las salidas requeridas, funcionalidad y característica del software. Estas historias deben de tener una respectiva prioridad la cual indicará a los programadores con qué

urgencia se debe de llevar a cabo el desarrollo de dicha historias. Si se estima que la historia requiere más de 3 semanas para desarrollarse, se le pide al cliente que la reduzca a historias más pequeñas.

  - Cuando se lleva a cabo la primera entrega del proyecto, se determina la velocidad de proyecto que no es más que la cantidad de historias realizadas en la presente iteración. La velocidad permite estimar tiempos de entrega para acordar las posteriores.

- Diseño
  - Se prefiere un diseño sencillo a una representación compleja.
  - Se usan tarjetas Clase-Responsabilidad-Colaborador (CRC) para situar el software en un entorno orientado a objetos.
  - Se propone el rediseño como técnica de optimización del diseño[1]. El rediseño puede ocurrir antes o después de la programación.
- Codificación
  - Antes de empezar a programar, se desarrollan pruebas unitarias de cada una de las historias que serán entregadas en la actual iteración. Estas pruebas permiten a los programadores centrarse en los aspectos que deben de cubrir para superar la prueba por parte de los *testers*.
  - La programación se lleva a cabo en pareja. El programador con menos experiencia es el encargado de codificar mientras que el más experimentado se encarga de buscar errores en el código además de cuestionar las decisiones hechas por el codificador con el fin de seguir los estándares de ingeniería de software. A su vez, esta práctica permite la diseminación de conocimiento entre ambos programadores.
  - Cuando se termina la labor de programar, se integra el código con el trabajo de los demás. Esta estrategia denominada como integración continua evade dificultades de comunicación y compatibilidad de interfaces además de que permite ir descubriendo errores en la marcha.
- Pruebas
  - Se llevan a cabo las pruebas, las cuales deben de usar una estructura que permita que puedan ser automatizadas con base a lo que el cliente requiera.
  - Las pruebas permiten al equipo darse cuenta si las cosas van en mal camino.
  - El cliente también participa en algo denominado como pruebas de aceptación las cuales son diseñadas por el cliente enfocándose en las funcionalidades y características del sistema. El cliente es apoyado por los *tester* para llevar a cabo las estas pruebas.

[1] Entiéndase rediseño como el proceso de mejorar la estructura interna y no alterar el comportamiento externo del código.

Metodología para usar en el presente proyecto

Una vez definido las 2 metodologías posibles a usar se considera lo siguiente

- Asesor que aporta conocimiento en el área de SPARQL, GeoSPARL, Web de *Linked Data* y contactos con personas involucradas en el desarrollo de Apache Marmotta.
- Asesora que aporta conocimiento sobre qué dirección debe de seguir el proyecto y qué requerimientos y restricciones debe de cumplir el proyecto para su entrega.
- Desarrolladores de Apache Marmotta, los cuales ya están en contacto con el desarrollador menos experimentado del proyecto, que proveerán de información y experiencia para el desarrollo.
- Desarrollador del proyecto quien estará en constante comunicación con los demás elementos del equipo para desarrollar el módulo de forma exitosa.

Con base a las características de ambas metodologías y las cualidades que cada elemento del equipo brinda, se considera que la metodología *Extreme Programming (XP)* es la adecuada para desarrollar el módulo de consultas federadas para el *triple store* Apache Marmotta.

Bibliografía

[1] E. S. Raymon, The Cathedral and the Bazaar, United States of America: Tim O'Reilly, 1999.

[2] A. Fuggetta, «Open source software—an evaluation,» *Journal of Systems and Software,* vol. 66, nº 1, pp. 77-90, 2003.

[3] B. Barry W., «A Spiral Model of Software Development and enhacement,» *Computer,* nº 5, pp. 61-72, 1988.

[4] T. Relihan, «MIT Sloan School,» MIT, 06 Julio 2018. [En línea]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/agile-scale-explained. [Último acceso: 06 Octubre 2019].

[5] B. Kent, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland y D. Thomas, «Manifiesto por el Desarrollo Ágil de Software,» Ward Cunningham, 6 Junio 2001. [En línea]. Available: http://agilemanifesto.org/iso/es/manifesto.html. [Último acceso: 6 Octubre 2019].

[6] J. R. Laínez Fuentes, Desarrollo de Software ÁGIL: Extreme Programming y Scrum, IT Campus Academy, 2015.