Capitulo 2. Marco Teórico

En el presente capítulo se describirán los conceptos utilizados para el desarrollo de ésta tesis. Entre los conceptos a mencionar se encuentran Patrones de Diseño, los Principios del Diseño Orientado a Objetos y "Test Driven Development".

2.1 Patrones de Diseño

Antes de definir lo que son los patrones de diseño, es necesario definir lo que un patrón significa. En base al diccionario de la real academia española un patrón puede ser definido como un modelo que sirve de muestra para sacar otra cosa igual.

Para el Arquitecto Christopher Alexander un patrón describe un problema que ocurre de manera frecuente además de describir la esencia de su solución para que pueda volver a ser usada [Alexander, 1979].

Los patrones de diseño son una serie de soluciones a problemas que se presentan de manera recurrente las cuales están basadas en experiencias anteriores. Según Allen Holub son una técnica general usada para resolver problemas de clases [Holub, 2004].

Los patrones de diseño tienen su origen en el año de 1979 cuando Christopher Alexander publica su libro "The Tímeles Way of Building" en el que describe una serie de problemas que se presentaron en la arquitectura y la solución a ellos había sido buena.

En 1987 Ward Cunninghan y Kent Beck presentaron en base a las ideas de Alexander un lenguaje de cinco patrones para iniciación en la programación en "Smalltalk", teniendo como marco el congreso "Object Oriented Programming Systems

and Lenguajes (OOPLSA)". A partir de 1994 se desarrollan dos conferencias al año especializada en patrones llamada "Patterns Lenguajes of Program Design" [García, 2005].

Más tarde en 1995 Gamma, Helm, Vlissides, y Johnson también conocidos como "Gang of Four" publican el libro "Design Patterns: Elements of Reusable Object-Oriented Software" en el que describen 23 patrones de diseño bajo el siguiente formato:

- Nombre del patrón y clasificación
- Propósito
- También conocido como
- Motivación
- Aplicabilidad
- Estructura
- Participantes
- Colaboraciones
- Consecuencias
- Implementación
- Código del ejemplo
- Usos Conocidos

- Patrones relacionados.

En el año 2004 Freeman, Freeman, Sierra y Bates publican el libro "Head First Design Patterns", en el cual de forma muy sencilla y con ejemplos prácticos se abordan los patrones de diseño.

Los patrones de diseño se clasifican en 3:

- Patrones de Creación.
- Patrones Estructurales.
- Patrones de Comportamiento.

Los patrones de creación son aquellos que se encargan de abstraer el proceso de instanciación de objetos [García, 2005]. Su objetivo es construir sistemas independientes sin importar la forma en que se crean, se componen o se representan los objetos. Los patrones que caen dentro de esta clasificación son:

- "Factory Method"
- "Abstract Factory"
- "Builder"
- "Prototype"
- "Singleton"

Los patrones estructurales se encargan de cuidar como las clases y objetos se componen para formar estructuras mayores [García, 2005]. Los patrones estructurales son:

- “Adapter”
- “Brigde”
- “Composite”
- “Decorador”
- “Facade”
- “Flyweight”
- “Proxy”

Los patrones de comportamiento son aquellos que se encargan de los algoritmos y las responsabilidades entre objetos, así como de la comunicación entre ellos [García 2005]. Los patrones de comportamiento según Gamma son:

- “Interpreter”
- “Template Method”
- “Chain of Responsability”
- “Command”
- “Iterator”

- “Mediator”
- “Memento”
- “Observer”
- “State”
- “Strategy”
- “Visitor”

Algunos de los principios del diseño orientado a objetos [Freeman, 2004] vinculados con los patrones de diseño son los siguientes:

- Identifica el aspecto que varía en la aplicación y sepáralo de la parte que no varía.
- Programa una interface y no una implementación.
- Favorece composición sobre herencia.
- Las clases deben estar abiertas para extensión y cerradas para modificación.
- Depender de abstracciones y no de clases concretas.
- No llames, nosotros te llamaremos.
- Una clase tiene solo una razón para cambiar.
- El principio de menos conocimiento llama solo a tus amigos cercanos.
- Esfuérzate por diseños débilmente acoplados entre objetos que interactúan.

2.2 “Test Driven Development”

La programación extrema aparece en los años 80’s y se define como un conjunto de reglas y prácticas que ocurren en el contexto de 4 actividades [Beck, 2005]:

- Planeación
- Diseño
- Codificación
- Pruebas

En la planeación se crean las hojas de usuario, las cuales describen las características y funcionalidad requerida para el software. A estas hojas de usuario se les asigna un valor por parte del cliente y un costo en semanas de desarrollo por parte los desarrolladores. Todas las hojas de usuario se deben implementar, sin embargo se da prioridad a aquellas con valor más alto o aquellas que son más riesgosas.

En el diseño se elaboran las tarjetas Colaborador-Responsabilidad-Clase que analizan y organizan las clases orientadas a objetos relevantes para el incremento actual. Si existe algún problema de difícil diseño, es en esta actividad en donde se deber crear un prototipo llamado solución pico.

Durante la codificación se llevan acabo 2 procesos, el refabricar (“refactoring”) y la programación en parejas.

Refabricar es el proceso de cambiar un sistema de tal forma que no altere su comportamiento externo pero que si mejore su estructura interna.

La programación en parejas es un proceso de integración continua, ya que 2 personas codifican en una misma computadora.

Las pruebas a realizarse son las siguientes:

Pruebas de unidad. Este tipo de pruebas son aquellas a realizarse a cada una de las historias de usuario, y se debe pensar que es lo que se va a implementar para que estas pruebas sean exitosas.

Pruebas de aceptación o pruebas del cliente. Estas pruebas son elementos visibles y revisables por el cliente, es decir las características generales y funcionalidad del sistema.

Pruebas de regresión. Éste tipo de pruebas se llevan a cabo cuando se agrega una nueva funcionalidad al software tanto como cuando se soluciona un error y sirve para verificar que el código nuevo no afecte el resto del sistema.

Conjunto Universal de pruebas. En este conjunto de pruebas podemos encontrar tanto a las pruebas de integración como a las pruebas de validación.

En la figura 2.2.1 se presentan las 4 actividades de la programación extrema.

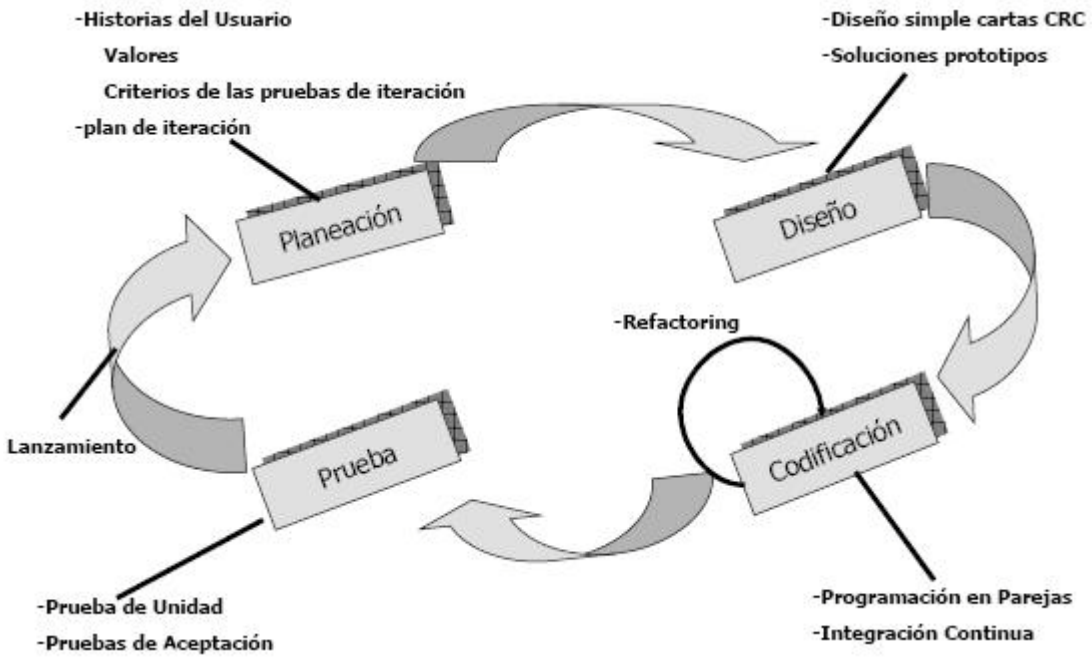

Figura 2.2.1 Actividades de Programación Extrema

Después de haber definido lo que es la programación extrema se puede definir lo que es “Test Driven Development”.

“Test Driven Development” es una técnica de programación extrema en la cual primero se desarrollan las pruebas de unidad de cualquier sistema, antes de escribir cualquier tipo de código funcional.

Ésta técnica es muy importante debido a que cada línea de código funcional pasa por un test, lo que ayuda a aislar fallos o errores en el sistema. Es importante mencionar que esta técnica solo aborda las pruebas de unidad, por lo que no deben dejarse a un lado el conjunto universal de pruebas, las pruebas de regresión y las pruebas de aceptación.