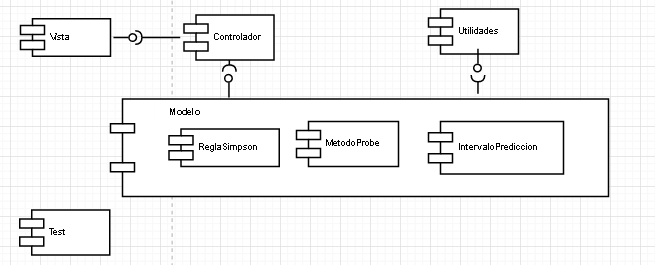
**Metaphor/Architecture Specification Template**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Walter Javier Alonso Roa | Date | 29-04-2017 |
| Program | Calculo intervalo de predicción | Program # | 7 |
| Instructor | Mario Riveros | Language | Java |

|  |  |
| --- | --- |
| **Design** | Functional Specification.rtf |
| **References** | Operational Scenarios.rtf |
|  | State Specification.rtf |
|  | Logic Specification.rtf |
|  |  |

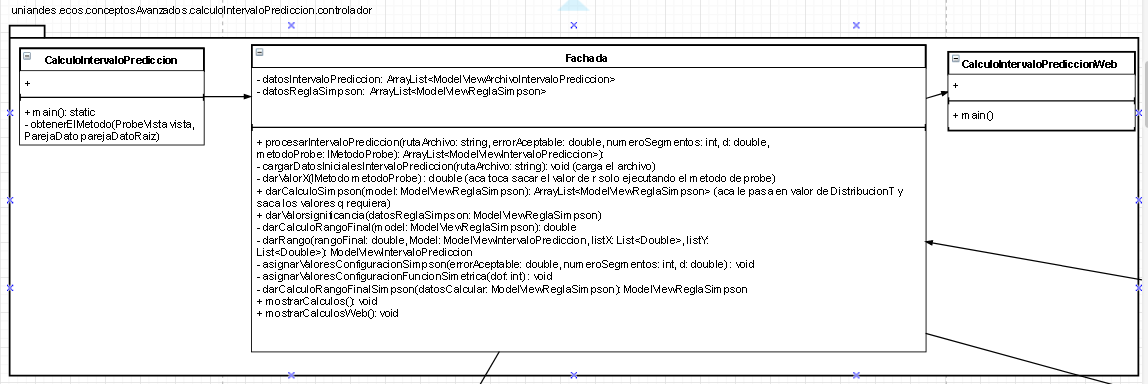
**Graphical representation of the metaphor**

**Diagrama de componentes:**



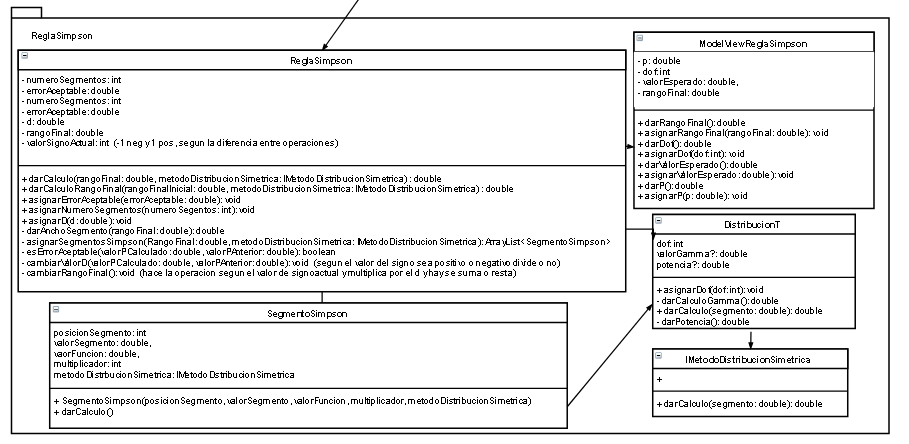
**Diagrama de clases:**

1. uniandes.ecos.conceptosAvanzados.calculoIntervaloPrediccion.controlador

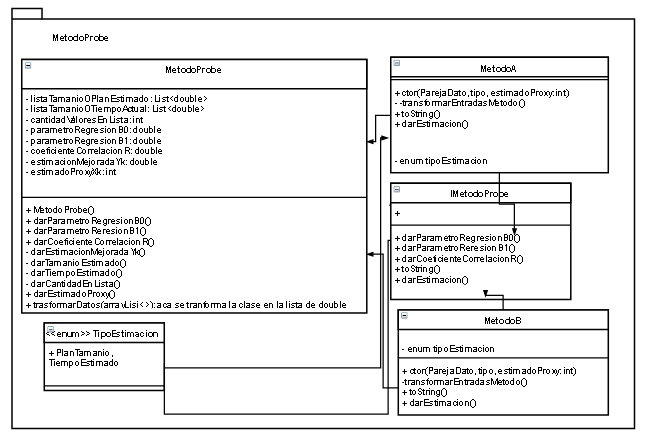


1. uniandes.ecos.conceptosAvanzados.calculoIntervaloPrediccion.modelo

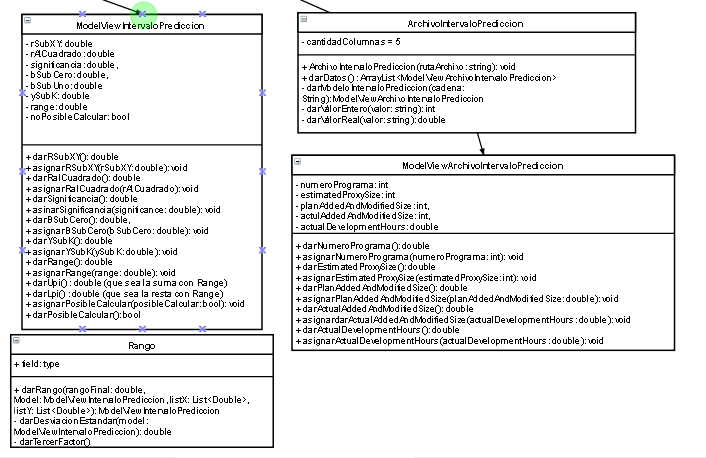
* ReglaSimpson



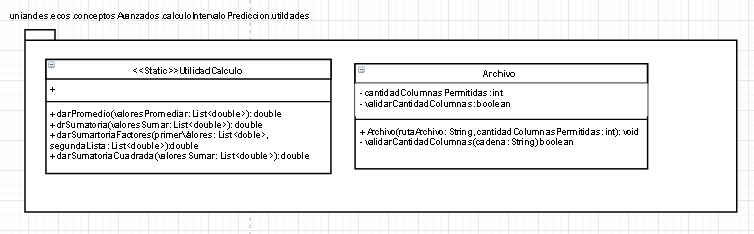
* Metodo Probe



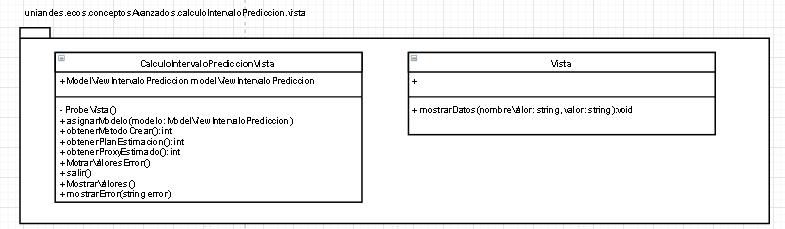
**Intervalo de prediccion:**



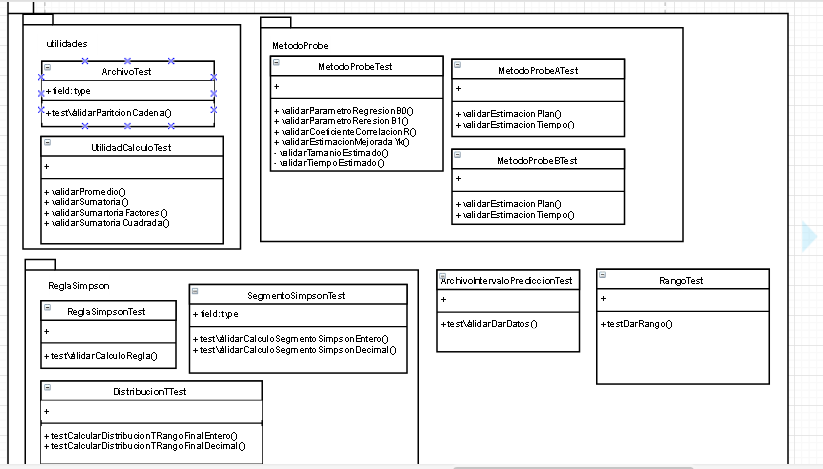
1. uniandes.ecos.conceptosAvanzados.calculoIntervaloPrediccion.utildades



1. uniandes.ecos.conceptosAvanzados.calculoIntervaloPrediccion.vista



1. uniandes.ecos.conceptosAvanzados.calculoIntervaloPrediccion.test



**Textual representation of metaphor**

Para el desarrollo de este sistema, se toman como componentes 3 programas desarrollados en programas pasados.

* Programa #3: Se crea el componente MetodoProbe, el cual ya tiene la lógica para realizar los cálculos de B0, B1, R y Yk.
* Programa #5 y #6: Se crea el componente ReglaSimpson, el cual tiene la lógica para obtener la información relacionada a los cálculos Simpson. Se clacularan los segmentos y valores aproximados para permitir obtener el rango.

**Los pasos del proceso son:**

1. El archivo con la información a procesar será manejada por las clases Archivo y ArchivoIntervaloPrediccion, las cuales tienen las reglas para verificar archivos correctos.
2. Por medio del patrón Fachada, la clase Fachada se encarga de orquestar toda la interacción desde la carga hasta el despliegue de información.
3. Las clases en ReglaSimpson (ReglaSimpson, SegmentoSimpson y DistribucionT), son las encargadas de indicar las operaciones que se deben realizar, estas clases usan la clase UtilidadCalculo, para realizar las operaciones necesarias.
4. Las clases en MetodoProbe (MetodoProbe,MetodoProbeA y MetodoProbeB), son las encargadas de indicar las operaciones que se deben realizar, estas clases usan la clase UtilidadCalculo, para realizar las operaciones necesarias.
5. DistribucionT implementa una interfaz llamada IMetodoDstribucionSimetrica, esto es porque ReglaSimpson es flexible en la fórmula que puede utilizar.
6. Para la visualizacion se tiene una clase llamada VistaIntervaloPredicción y Vista que se encargan de mostrar en consola la información y un archivo intervaloPredicción.tfl, el cual aunque no es Java contiene código Html y el cual se utilizara para desplegar en Heroku.
7. Se tienen varias clases a las cuales se van a generar test, dado que son clases lógicas:

* ArchivoTest.
* ReglaSimpsonTest.
* ArchivoIntervaloPrediccionTest
* SegmentoSimpsonTest
* UtilidadCalculoTest
* DistribucionTTest
* MetodoProbeTest
* MetodoProbeATest
* MetodoProbeBTest

**Decisiones técnicas:**

* Se utiliza el patrón fachada con el objetivo que coordine el proceso de carga y despliegue de información.
* Se utiliza una un patrón de arquitetura MVC, con el objetivo de separar la lógica del sistema y datos.
* Regla Simpson es manejado como un Simgleton, con el objetivo de tener una sola instancia de ReglaSimpson en el sistema.
* Para realizar el caculo core del sistema, se tienen 2 componentes con las clases principales del proceso:
* ReglaSimpson : ReglaSimpson, SegmentoSimpson y DistribucionT.
* MetodoProbe: MetodoProbe, MetodoProbeA y MetodoProbeB.

Ademas de los compoentes este programa tiene otras clases para cumplir con lo deseado:

Rango

* Se implementará una arquitectura por capas con la idea de separar lógica del sistema y sea más fácil su mantenimiento, además de dejarlo en componentes.
* Se usa la librería Spark con el objetivo de desplegar información web.
* Se usa la librería JUnit para realizar pruebas unitarias.
* Se desplegará en Heroku la aplicación, así como en consola.

**Para este programa #7**

Se mantiene la estructura de proceso y decisiones técnicas del programa #6 y #5.

Como nuevo, se debe agregar funcionalidad en el cálculo del rango para que devuelva este valor según requerimiento y especificación del requerimiento.

Metaphor/Architecture Specification Template Instructions

|  |  |
| --- | --- |
| Purpose | * To contain the metaphor for a program, component, or system * To enable precise, rapid and complete design understanding * To facilitate thorough design and implementation reviews and inspections |
| General | * Use this template to document the program’s high-level metaphor. * The metaphor could be based in common programming patterns as MVC, or architectural styles as tree layer design, client-server, or inversion of control frameworks * After implementation and testing, update the template to reflect the actual implemented product. * Use plain language and avoid using programming instructions wherever practical. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Design References | List the references used to produce the program’s design.   * the Operational, Functional, and State templates * the program’s requirements * any other pertinent source |
| Graphical representation of the metaphor/Architecture | * Create a graphical representation of the main program parts and its interactions * Use clear names for each part * Use edges with arrows to show interactions * Use descriptive names for the interactions |
| Textual representation of metaphor | * Use text to describe the main idea and metaphor used in your design * Describe the graphical representation using common language |