**Software Design Document**

**(SDD)**

**Presentado a:**

Ing. Mariela Curiel.

David Calle

Alfredo Santamaría

David Suárez

Contenido

[1 Introducción 4](#_Toc467012768)

[1.1 Descripción de la Audiencia 4](#_Toc467012769)

[1.2 Descripción del Documento 4](#_Toc467012770)

[2 Arquitectura 5](#_Toc467012771)

[2.1 Vista Lógica del Sistema 5](#_Toc467012772)

[2.1.1 Generador de Trabajos 5](#_Toc467012773)

[2.1.2 Asimilador de Trabajos 6](#_Toc467012774)

[2.2 Vista Física del Sistema 7](#_Toc467012775)

[2.3 Vista de Procesos del Sistema 9](#_Toc467012776)

[2.3.1 Generador de Trabajos 9](#_Toc467012777)

[2.3.2 Asimilador de Trabajos 9](#_Toc467012778)

[3 Diseño Detallado 10](#_Toc467012779)

[3.1 Generador de Trabajos 10](#_Toc467012780)

[3.1.1 Boinc Integration 10](#_Toc467012781)

[3.2 Asimilador de Trabajos 11](#_Toc467012782)

[3.2.1 ConcreteAssimilator 11](#_Toc467012783)

[Bibliografía 11](#_Toc467012784)

Figuras

[Figura 1 Estructura del Documento 5](#_Toc467012785)

[Figura 2 Diagrama de Clases Generador de Trabajos 6](#_Toc467012786)

[Figura 3 Diagrama de Clases Asimilador de Trabajos 7](#_Toc467012787)

[Figura 4 Arquitectura de Boinc. 8](#_Toc467012788)

[Figura 5 Diagrama de procesos Generador de Trabajos 9](#_Toc467012789)

[Figura 6 Diagrama de Procesos Asimilador de Trabajos 10](#_Toc467012790)

# 1 Introducción

## Descripción de la Audiencia

El presente documento está dirigido a cualquier persona que desee continuar o aprender sobre la investigación de Grids Móviles de la Pontificia Universidad Javeriana. El objetivo es plasmar los diseños hechos por el grupo, explicar cómo estos se integran con Boinc y los dispositivos móviles con sistema operativo Android. Además, este documento le puede interesar a un programador que requiera integrar sus algoritmos y desee utilizar los componentes desarrollados en este trabajo de grado. Es importante resaltar que se espera que el interesado esté familiarizado con el funcionamiento de Boinc ya que se sale del alcance de este documento describirlo a detalle. Sin embargo, esta información se puede encontrar en la propuesta del trabajo de grado.

## Descripción del Documento

En esta sección se describe la estructura del documento SDD (Software Design Description) especificando la distribución de cada una de sus partes. Lo anteriormente mencionado se encuentra descrito en la siguiente figura.

Figura 1 Estructura del Documento

# 2 Arquitectura

## 2.1 Vista Lógica del Sistema

Según el modelo 4 + 1 la vista lógica está relacionada con la funcionalidad que el sistema provee a los usuarios finales [1]. Dado que el usuario final es el programador que requiere adaptar sus algoritmos a la grid, en esta vista se describen los componentes que permiten lograr este objetivo. En ese orden de ideas se describirán 3 partes del sistema.

### 2.1.1 Generador de Trabajos

El primero es el generador de trabajos, este componente implementa el patrón adaptador [2], permitiendo integrar un algoritmo de división previamente implementado con Boinc y crear trabajos sin que el programador conozca los detalles específicos del funcionamiento. En este diagrama se puede notar la presencia de la clase WorkDivisor, esta clase representa a la fachada [3] de un algoritmo de división previamente implementado. Esta fachada se comunica con el adaptador y crea un trabajo dada una lista de rutas de archivos. El adaptador se encarga de comunicarse con el api del servidor y retorna el identificador del trabajo creado. Esto le servirá al programador en el futuro para poder unir sus trabajos.

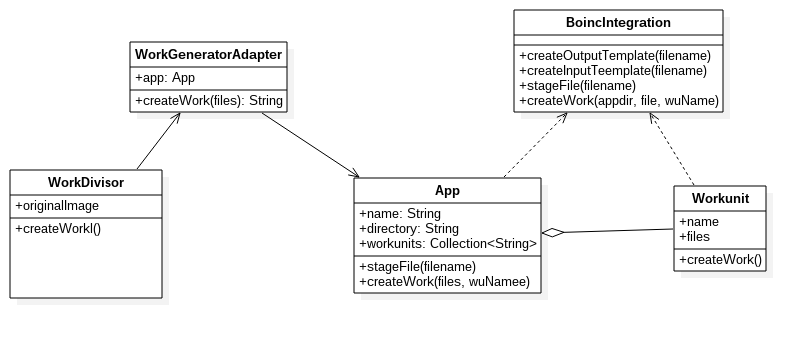


Figura 2 Diagrama de Clases Generador de Trabajos

### 2.1.2 Asimilador de Trabajos

El segundo componente desarrollado fue el asimilador de trabajos. El objetivo de este componente es hacer operaciones con los resultados de las unidades de trabajo terminadas, ya que una vez termina un trabajo y es validado, Boinc se encarga de borrar todos los archivos subidos por el cliente. Esto hace que sea necesario usar los resultados de las operaciones inmediatamente o guardarlos de tal forma que no haya pérdida de información.

Debido que existen múltiples algoritmos para unir imágenes dependiendo de la aplicación específica, se decidió crear un asimilador que fuese lo suficientemente extensible para que dado uno o muchos algoritmos de unión previamente implementados, estos puedan ser adaptados al proyecto de la manera más fácil posible. Para cumplir este requerimiento se hizo una implementación del patrón observador, donde los observadores son los algoritmos de unión implementados por el programador y el observado es una clase que se encarga de detectar cuando una unidad de trabajo está lista para asimilar y de notificar los cambios para que puedan ser procesados. Es deber del programador implementar la interfaz del observador y registrarse ante el observado.

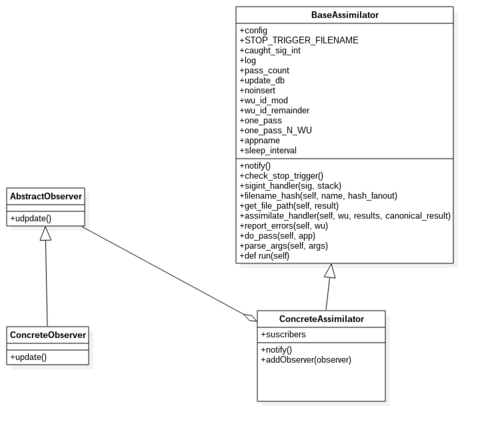


Figura 3 Diagrama de Clases Asimilador de Trabajos

## 2.2 Vista Física del Sistema

Boinc es una aplicación cliente servidor que corre sobre el servidor web apache. En general el despliegue hace referencia a dos nodos principales, el cliente y el servidor. El servidor es un vps con sistema operativo Ubuntu 14.04 LTS, sobre este está instalado Apache v2.4.7 y MySQL 5.7. Dentro de Apache se corren varios procesos en segundo plano de Boinc cada uno con una funcionalidad diferente. El primero es el feeder, este se encarga de recibir las peticiones de trabajo del cliente y enviárselas, el segundo es el validador, el cual una vez un cliente ejecuta el trabajo y lo sube, se encarga de revisar si el resultado es correcto. En caso de que sea correcto, el transitioner les cambia el estado a las unidades de trabajo; cuando una unidad de trabajo cambia al estado listo para asimilar, el asimilador obtiene los resultados y aplica operaciones sobre los mismos. Por último, una vez el asimilador haya procesado el resultado y con el objetivo de no desperdiciar almacenamiento en el servidor, se ejecuta un componente que se encarga de eliminar los archivos que ya no son necesarios, llamado file deleter.

La segunda parte del sistema hace referencia al cliente, que en este caso es un teléfono Android con sistema operativo mayor o igual a 4.1. Dentro de este teléfono está instalado la aplicación Boinc. Esta se encarga de pedirle trabajos al servidor que tiene dos ejecutables principales, el wrapper de Android y la aplicación de ITK. El wrapper se encarga entonces de ejecutar la aplicación de ITK y comunicarle los resultados al servidor.

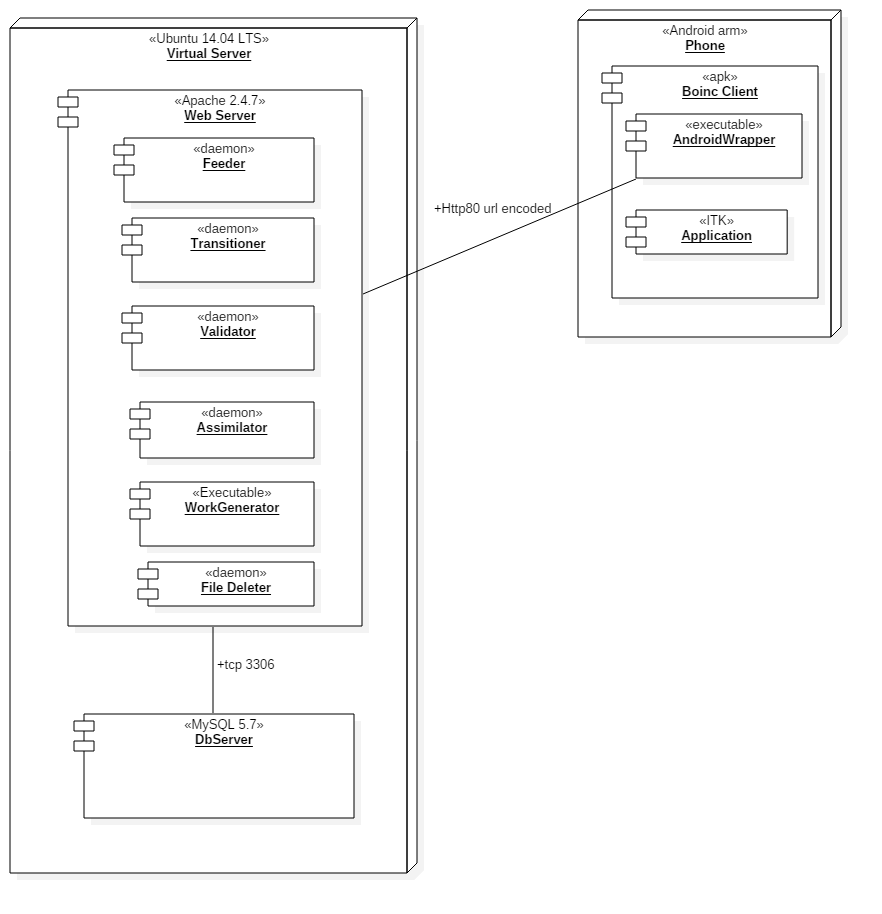


Figura 4 Arquitectura de Boinc.

## 2.3 Vista de Procesos del Sistema

En esta sección se hará una descripción de los procesos del sistema, de los componentes desarrollados a lo largo de este trabajo de grado. Es decir, el asimilador y el generador de trabajos. Como prerrequisito es necesario haber configurado el servidor Boinc y haber creado tanto un proyecto como una aplicación.

### 2.3.1 Generador de Trabajos

El flujo de trabajo principal del generador es el siguiente. El código creado por el programador, que divide la imagen en partes, utiliza el adaptador comunicándole que debe crear uno o varios trabajos a Boinc con determinados archivos de entrada. El adaptador se encarga de comunicarse directamente con el api de Boinc e inicializa los trabajos con un identificador único.

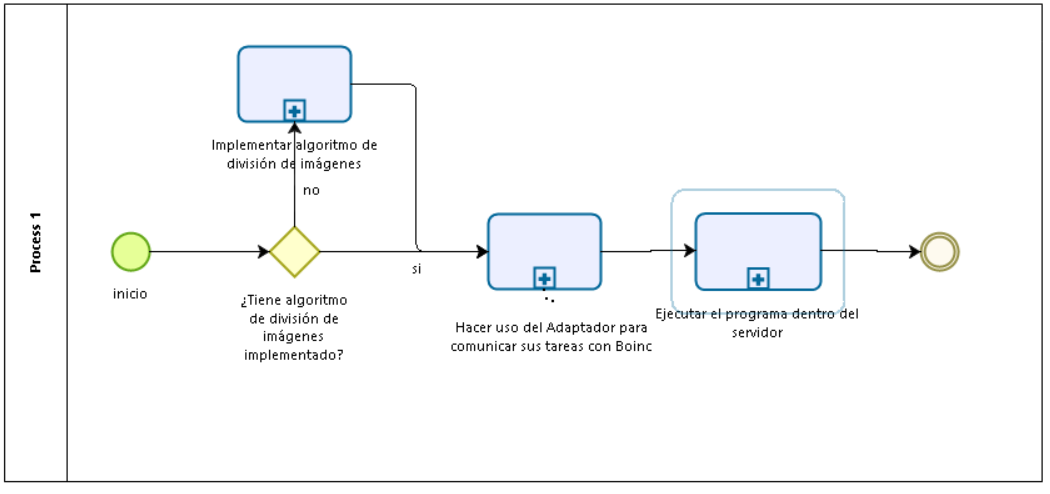


Figura 5 Diagrama de procesos Generador de Trabajos

### 2.3.2 Asimilador de Trabajos

El flujo del asimilador es el siguiente. El código desarrollado por el programador se registra a el observador en busca de conocer las tareas terminadas y sus resultados. El observado utiliza el api de Boinc para buscar las tareas terminadas; si encuentra nuevas tareas para asimilar, notifica a todos los observadores y le cambia el estado a la tarea a asimilada.

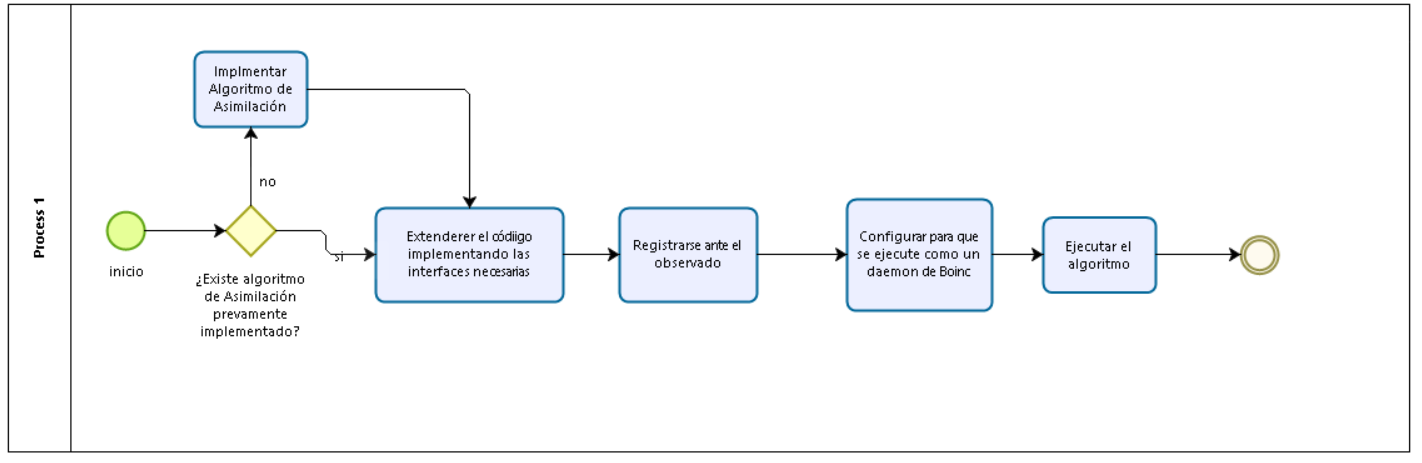


Figura 6 Diagrama de Procesos Asimilador de Trabajos

# 3 Diseño Detallado

## 3.1 Generador de Trabajos

El generador de trabajos consiste en dos partes principales, el primero es el BoincIntegration, este componente consiste en un conjunto de clases que permite comunicarse con el api de Boinc. El segundo es el adaptador, el objetivo de este es que el programador pueda crear trabajos en Boinc sin necesidad de tener conocimientos avanzados de la comunicación con Boinc. En este paso solo se necesita el nombre de la aplicación y el adaptador se encarga de inicializar los objetos necesarios para crear trabajos.

### 3.1.1 Boinc Integration

Esta parte se comporta como un wrapper y tiene 3 clases principales

* BoincIntegration
  + Wrapper directo al api de Boinc. Permite crear tareas, agregar archivos a la carpeta de descargas y generar templates de entrada y de salida.
* App
  + Clase encargada de guardar información sobre los Workunits creados, la aplicación actual y de comunicarse con BoincIntegration para crear los trabajos.
* Workunit
  + Clase que representa una unidad de trabajo en Boinc, guarda el identificador de la tarea y los archivos de entrada de esta tarea.

## 3.2 Asimilador de Trabajos

Esta parte del sistema es una extensión de lo que provee Boinc actualmente (BaseAssimilator). El objetivo es que el programador solo sepa los resultados de las tareas y no tenga que hacer operaciones manuales para que Boinc funcione correctamente

### 3.2.1 ConcreteAssimilator

El asimilador consiste en 3 partes principales

* ConcreteAssimilator
  + Se comporta como el componente observado del patrón observador y guarda un conjunto de suscriptores, los observadores, a los cuales notifica cada vez que acaba una tarea.
* AbstractObserver
  + Clase abstracta cuyo objetivo es hacer que el programador implemente en su asimilador
  + Lo único que pide es que sus hijos implementen el método update para que pueda ser llamado por el ConcreteAssimilator.
* ConcreteObserver
  + Esta clase debe ser implementada por el programador que conoce la aplicación específica. Su objetivo es tomar los resultados independientes y realizar las operaciones necesarias para su asimilación.

# Bibliografía

|  |  |
| --- | --- |
| [1] | S. Systems, «SparksSistems,» [En línea]. Available: http://www.sparxsystems.com/downloads/whitepapers/FCGSS\_US\_WP\_Applying\_4+1\_w\_UML2.pdf. |
| [2] | Microsoft, «Microsoft,» [En línea]. Available: https://msdn.microsoft.com/en-us/library/orm-9780596527730-01-04.aspx. |
| [3] | SOA, «SOA Patterns,» [En línea]. Available: http://soapatterns.org/design\_patterns/service\_facade. |