Proyecto Final

Estefana Garca Gonzlez, Sebastin Mora Sabogal

26 de mayo de 2019

Índice general

Índice de figuras

Parte I

PROYECTO

Capítulo 1

Caso de Estudio

1.1. Introduccin

Desde que el ser humano cuenta con raciocinio , ha buscado organizarse, desarrollar metodologas y nuevas tecnologas que faciliten su diario vivir. Ha habido un recorrido histrico en el cual las necesidades humanas de optimizacin de tiempo y recursos han ido en aumento, as mismo las soluciones a stas. En los ltimos aos se ha podido apreciar una constante migracin al uso de tecnologas de la informacin que permiten realizar a cabo tareas en todos los mbitos de forma ptima. Uno de los actores que ms se han visto inmersos en la revolucin digital son los estudiantes, pero en su contexto universitario, hace falta desarrollar estrategias que le permitan mejorar la gestin de tiempo de sus actividades acadmicas; por lo cual se buscar una solucin tecnolgica que se adapte a las necesidades de los universitarios.

1.2. Objetivo General

Desarrollar un software que gestione actividades y tiempos de las asignaciones acadmicas a estudiantes universitarios, utilizando los modelos y metodologas de ingeniera de software para mejorar la productividad del universitario.

1.3. Objetivos Especficos

1. Analizar el problema teniendo en cuenta la observacin de las necesidades del estudiante, para as enfocarse en estos elementos primordiales a la hora de desarrollar el software.

2. Presentar una solucin a nivel de software a partir del previo anlisis del problema para finalmente implementarlo.

1.4. Descripcin del problema

La vida universitaria y acadmica suele ser difcil de manejar debido a la cantidad de trabajos que se deben entregar diariamente, a la prioridad que cada una es para el usuario y a la gestin de tiempo para poder realizarlos. Tareas, trabajos, talleres y grandes proyectos son algunas de las actividades que un estudiante realiza durante su semestre; adems de que cada uno tiene complejidad y tiempo de realizacin diferentes estimados por el estudiante. Una solucin factible es la utilizacin de un software gestor de tareas orientado a la organizacin y optimizacin de actividades acadmicas.

1.5. Alcance

Este software tendr la capacidad de gestionar los horarios de los estudiantes, aadir recordatorios de trabajos prximos a presentar y ofrecer el servicio de organizar en horarios la realizacin de las tareas pendientes. Esto se llevar a cabo de acuerdo a la complejidad de la actividad a realizar, en la cual se tomar en cuenta el nivel de dificultad, si se puede desarrollar en diferentes etapas y la fecha de entrega.

El estudiante estar en la capacidad de aadir actividades, determinar la complejidad de stas y asignarles un horario de realizacin que puede ser repartido en varios bloques cuando la tarea requiere de mucho tiempo. Adicionalmente, las actividades podrn personalizarse aadiendoles objetivos a cumplir o subactividades.

Capítulo 2

Metodologa

2.1. Introduccin

La metodologa del proceso de software que se debe seguir, es fundamental pues define las acciones generales que se deben llevar, a modo de conseguir un desarrollo del proyecto optimo, pasando por cada una de las fases del proceso elegido.

2.2. Proceso de Software

Parte importante de un proyecto de software es definir el, o los ciclos de vida que se manejarn dentro del proyecto, ya que estos determinarn estrategias para planificar,desarrollar y mantener el software. Por esta razn, se definir el modelo de procesos a utilizar, tomando en cuenta los siguientes criterios:

- Es necesaria una metodologa que sea pertinente para un proyecto de software pequeo con pocos desarrolladores.
- Se considera importante la verificacin en cada fase del ciclo de vida, ya que permite sentar buenas bases dentro del proyecto y reducir el riesgo.
- Adems de la verificacin, es necesaria una retroalimentacin constante, ya que es posible ver con mayor claridad las falencias y carencias del proyecto.
- Como ltimo criterio fundamental, se contempla la necesidad de desarrollar algunas partes de software de forma rpida, ya que esto facilitara

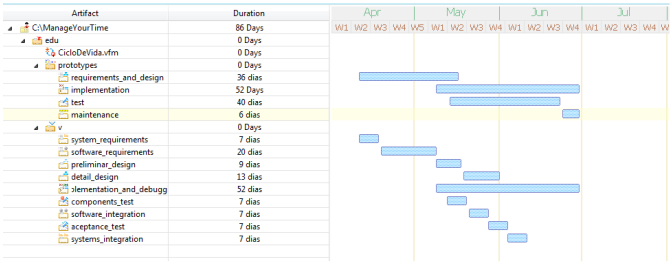

Figura 2.1: Cronograma. Diagrama de Gantt

la retroalimentacin del sistema.

Para cumplir con las pautas anteriormente mencionadas, los ciclos de vida que se elegirn son prototipo y V. Cada uno de estos modelos obedece solo a algunas de las especificaciones, pero juntos se complementan de la siguiente manera:

- El modelo V es perfecto para equipos de trabajo pequeos, ya que es sencillo, de fcil aprendizaje, robusto e incluye pruebas en cada fase, lo que facilita el trabajo cuando hay pocas personas.

- Gracias a los dos ciclos de vida, es posible hacer una verificacin y retroalimentacin de forma efectiva, ya que con el modelo en V se hacen pruebas en cada fase y con el prototipo es posible obtener resultados a corto plazo que se pueden ir revisando y evaluando.

- El modelo de prototipo brinda la posibilidad de construir partes del proyecto de forma prematura, por lo que es posible realizar pruebas y verificar qu cosas es necesario cambiar o aadir.

2.2.1. Metodologa de implementacin

Los criterios que se establecieron al momento de justificar la eleccin los procesos de software prototipo y V, cuentan con la misma validez para determinar la metodologa de implementacin, debido a que para esta etapa tambin es necesario tener un plan de accin que beneficie la gestin de tiempos del proyecto, complemente los procesos de software, cumpliendo un proceder de forma organizada. Por esta razn se utilizar Scrum como metodologa para implementar.

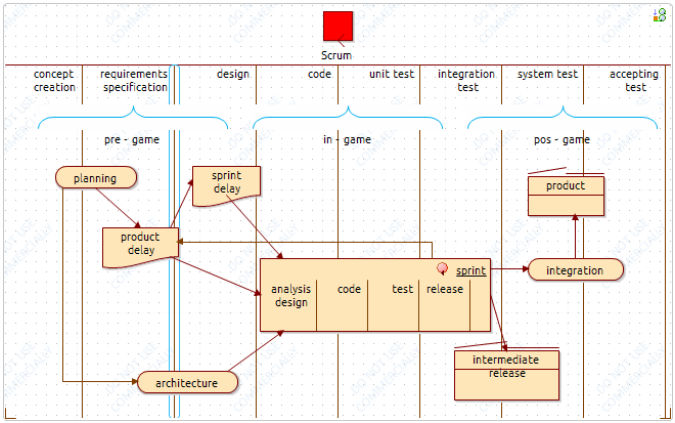

Figura 2.2: Scrum

2.3. Open Source

Desde que las personas empezaron a desarrollar software, han empezado a indagar en diferentes formas de realizar las cosas, a fin de obtener la solucin computacional que solucione su necesidad. Con el tiempo estos pensamientos han devenido en ideologas que orientan la variedad de metodologas disponibles para desarrollar software.

El pensamiento o filosofa que entra en cuestin, es la del software libre, donde uno de sus principios, consiste en la reutilizacin del conocimiento, en este caso, el cdigo. Es aqu donde entra el Open Source, que se relaciona con el cdigo abierto, y con su revisin por parte de una comunidad de desarrolladores externos. Siguiendo el principio de filosofa libre, se pretende utilizar el concepto O.S con la intencin de obtener una ayuda en momentos donde la implementacin se torne complicada, llegandose a extrapolar a diversos casos en los que se necesite la apreciacin del problema que se est trabajando por parte de un externo el cual ya lo haya desarrollado.

Parte II

DISEO

Capítulo 3

Requerimientos

3.1. Introduccin

Para cualquier proyecto de software, es un punto fundamental conocer cul es la necesidad y el problema que el cliente desea resolver. Para tener una visin holstica del problema, se hace necesario definir los requerimientos que satisfagan al cliente y resuelvan el problema.

3.2. Requerimientos del Cliente

Se entiende como lo que el cliente espera encontrar cuando interacte con la aplicacin. Bajo la anterior premisa, se definieron los siguientes requerimientos:

1. Aadir una tarea.
2. Aadir subtareas para una tarea.
3. Aadir un horario universitario.
4. Aadir un horario de descanso (dormir).
5. Aadir un horario de transporte.
6. Aadir una tarea a una materia.
7. Mostrar todas las tareas pendientes.
8. Mostrar las tareas pendientes por materia.
9. Mostrar las tareas pendientes por tipo.

10. Mostrar las tareas pendientes para una fecha.

11. Mostrar las tareas pendientes por dificultad.

12. Mostrar el horario general del usuario.

13. Mostrar los horarios asignados para las tareas pendientes.

14. Modificar horario.

15. Modificar tarea.

16. Sugerir horarios para realizar tareas.

17. Sugerir cunto tiempo podra tomar una tarea.

18. Sugerir tiempos de pausas activas durante la realizacin de una tarea.

19. Alertar de la prxima entrega de una tarea.

20. Advertir si se debe sacrificar algn espacio de descanso.

3.3. Casos de uso

Los casos de uso describen la interaccin del usuario con las diversas funcionalidades planteadas, permitiendo obtener una forma de comunicar los requerimientos de tal forma que sea entendida tanto por usuario como por desarrolladores. Como se podr observar a continuacin, sern cuatro diagramas de caso de uso los que se presentan, donde el motivo por el cual los casos de uso comparten diagrama, es porque se considera que existe cierta relacin entre ellos.

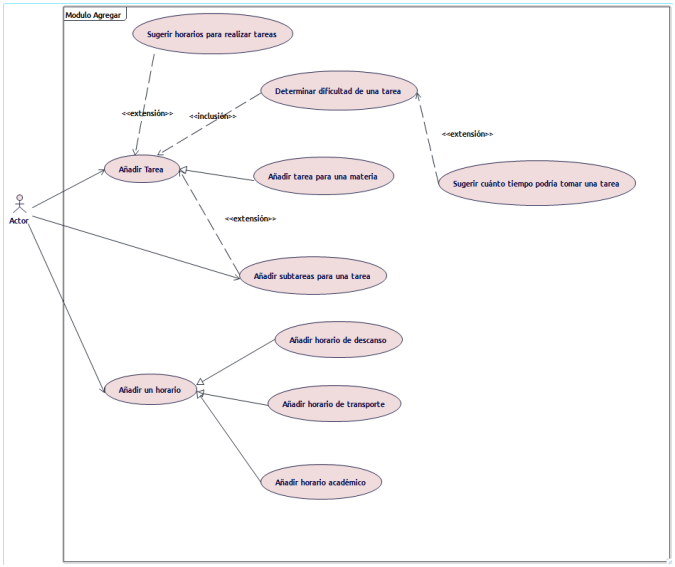

Figura 3.1: Primer diagrama de caso de uso

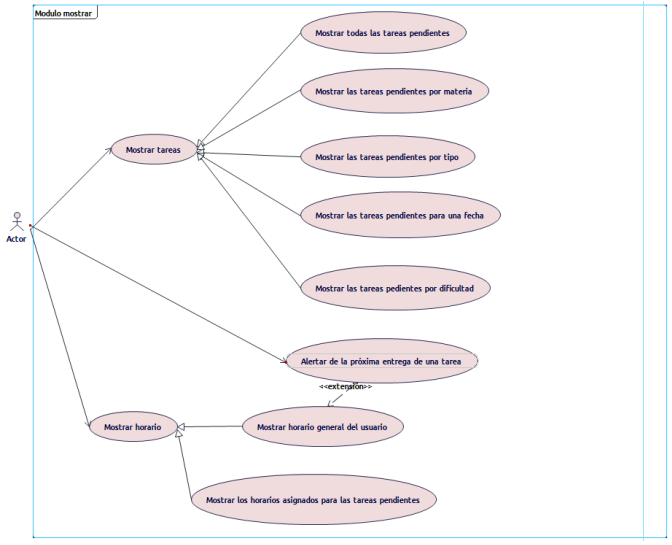

Figura 3.2: Segundo diagrama de caso de uso

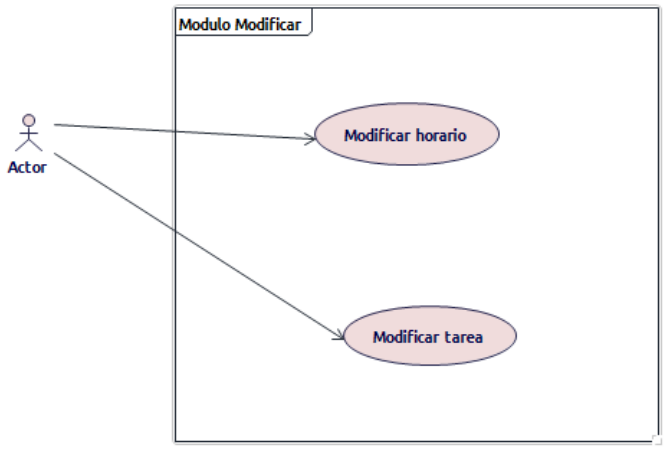

Figura 3.3: Tercero diagrama de caso de uso

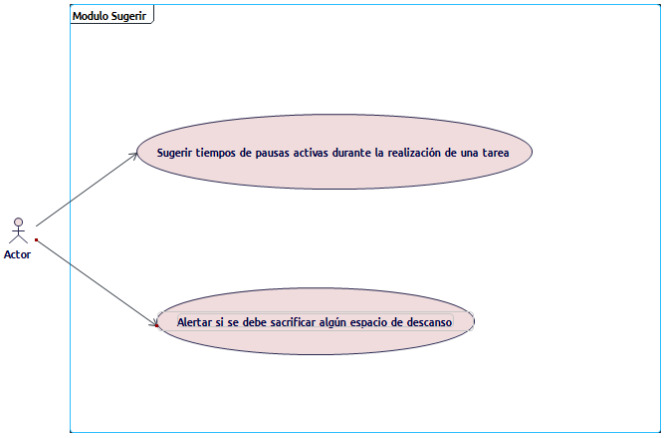

Figura 3.4: Cuarto diagrama de caso de uso

Los diagramas a continuacin representan gran importancia complemetando la definicin de los requerimientos.

3.4. Diagramas de secuencia

Los diagramas de secuencia permiten obervar la realizacin del caso de uso, responden el como se va a hacer el requerimiento. Los siguientes son los diagramas de secuencia de 4 casos de uso que se consideran de mayor importancia.

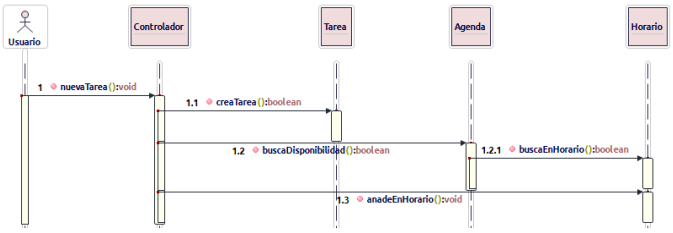

Figura 3.5: Diagrama de secuencia caso de uso 01.

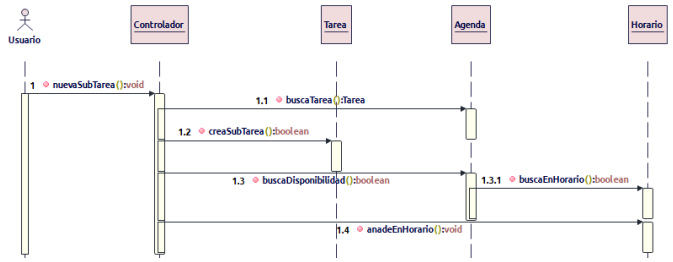

Figura 3.6: Diagrama de secuencia caso de uso 02.

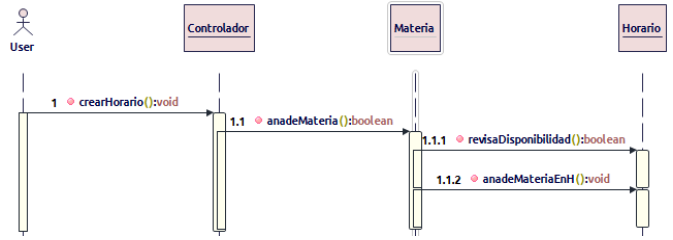

Figura 3.7: Diagrama de secuencia caso de uso 03.

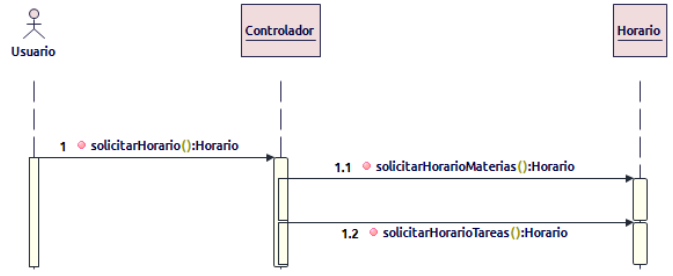

Figura 3.8: Diagrama de secuencia caso de uso 05.

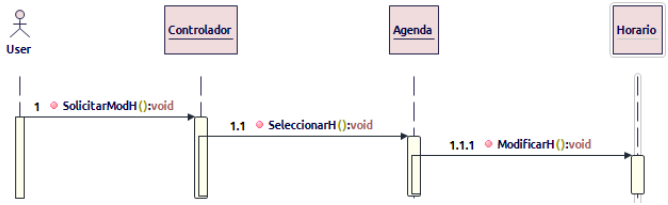

Figura 3.9: Diagrama de secuencia caso de uso 14.

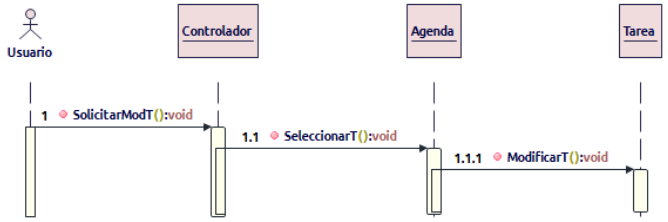

Figura 3.10: Diagrama de secuencia caso de uso 15.

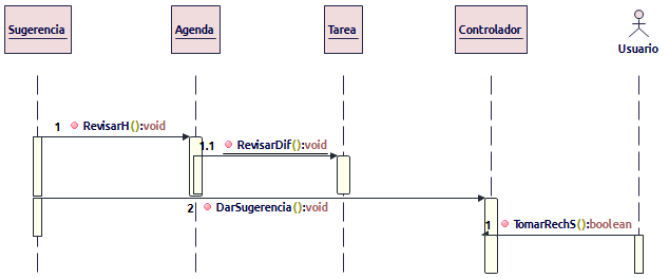

Figura 3.11: Diagrama de secuencia caso de uso 16.

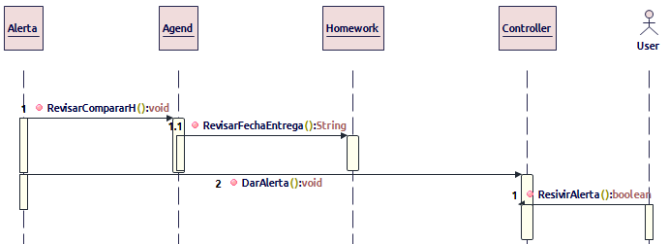

Figura 3.12: Diagrama de secuencia caso de uso 19.

3.5. Diagramas de comunicacin

Igualmente relacionados con los diagramas anteriores, principalmente con el diagrama de secuencia. Su funcin como su nombre lo indica, consiste en detallar en como se comunican los objetos que solucionan el requerimiento.

Se presentan los diagramas correspondientes a los ya expuestos diagramas de secuencia:

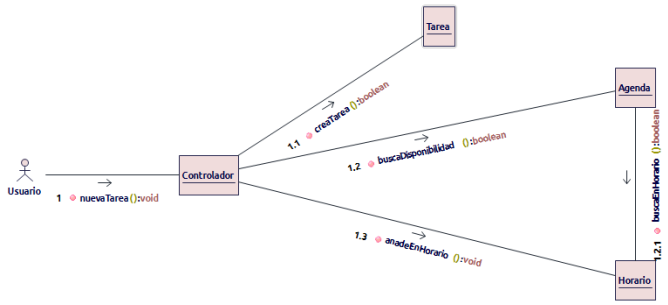

Figura 3.13: Diagrama de comunicacin caso de uso 01.

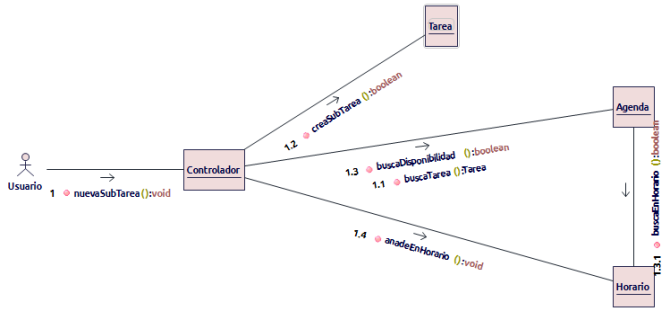

Figura 3.14: Diagrama de comunicacin caso de uso 02.

Figura 3.15: Diagrama de comunicacin caso de uso 03.

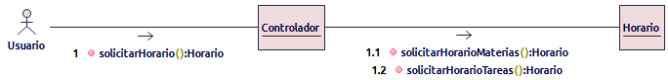

Figura 3.16: Diagrama de comunicacin caso de uso 05.

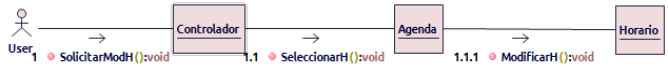

Figura 3.17: Diagrama de comunicacin caso de uso 14.

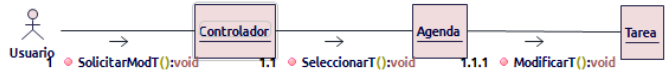

Figura 3.18: Diagrama de comunicacin caso de uso 15.

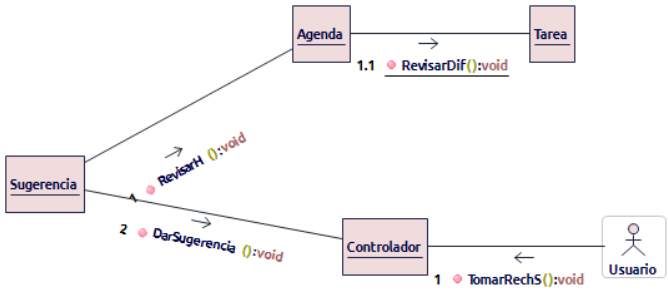

Figura 3.19: Diagrama de comunicacin caso de uso 16.

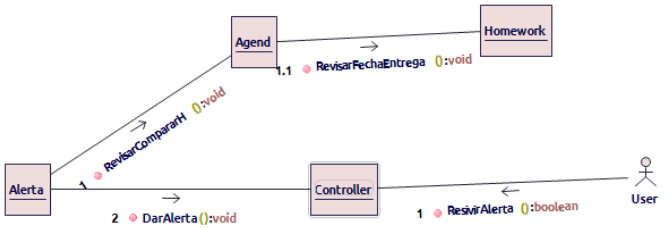

Figura 3.20: Diagrama de comunicacin caso de uso 19.

Las siguientes tablas, son la especificacin de los requerimientos que se considera que tienen un nivel de importancia alta.

RF-01	Aadir una tarea	
Descripcin	El usuario aade una tarea pendiente por desarrollar.	
Precondicin	El usuario debe tener un horario	
Secuencia	Paso	Accin
	1	El usuario selecciona la opcin de crear tarea.
	2	El usuario proporciona la informacin requerida (nombre de la tarea, tipo, materia a la que pertenece)
	3	El usuario verifica la informacin registrada.
	4	El usuario hace selecciona el botn aceptar.
Postcondicin	El sistema muestra la tarea recin asignada con sus especificaciones y su recomendacin de tiempo de realizacin y de horario	
Excepciones	Paso	Accin
	4	Se aade una tarea que requiere urgencia (Imprevisto). El usuario elige que horario sacrificar para realizar la tarea.
	4	Se aade una tarea que es imposible de realizar debido al tiempo u horario. Es necesario modificar los tiempos u horarios en los que se realizar la tarea o elegir si sacrificar una frnaja de horario.
Rendimiento	Paso	Cota de tiempo
	1	1 segundo
	2	40 segundos
	3	5 segundos
	4	1 segundo
Importancia	Muy importante	
Urgencia	urgente	

RF-02	Aadir subtareas para una tarea.	
Descripcion	Se aade una subtarea a una tarea.	
Precondicin	Debe existir alguna tarea pendiente.	
Secuencia	Paso	Accin
	1	El usuario selecciona la tarea a la que desea aadirle una subtarea.
	2	Seleccionar la opcin de aadir subtarea.
	3	Se aade la subtarea como una tarea (RF-01)
	4	El usuario verifica la informacin.
	5	El pulsa la opcin de aceptar.
Postcondicin	El sistema aadir la subtarea a la tarea, mostrar sus especificaciones y recomendacin de tiempo de realizacin y de horario	
Excepciones	Paso	Accin
	1	No existe una tarea para aadirle una subtarea.
	3	La subtarea es de carcter urgente.
	4	Se aade una subtarea y esta hace que la tarea sea imposible de terminar debido al tiempo u horario.
Rendimiento	Paso	Cota de tiempo
	1	5 segundos
	2	1 segundo
	3	40 segundos
	4	5 segundos
	5	1 segundo
Importancia	Importante	
Urgencia	No urgente	
Comentarios	No.	Descripcin
	1	Aadir una subtarea es lo mismo que aadir una tarea, la deferencia es que est anidada dentro de una tarea general.

RF-03	Aadir un horario universitario	
Descripcin	Se crea un horario con materias de la universidad.	
Precondicin	Ser un usuario registrado.	
Secuencia	Paso	Accin
	1	Seleccionar la opcin de crear horario.
	2	Escribir el nombre de cada materia y su respectiva hora de inicio y fin y los das en que se repite.
	3	El usuario aade la materia y repite el proceso cuantas veces sea necesario.
	4	Pulsar en el botn de aceptar.
Postcondicin	El sistema guardar el horario asignado para el usuario.	
Excepciones	Paso	Accin
	3	El horario de la universidad llena totalmente los espacios disponibles.
	3	No hay espacios disponibles para aadir ms materias al horario.
Rendimiento	Paso	Cota de tiempo
	1	1 segundo
	2	40 segundos
	3	2 minutos
	4	1 segundo
Importancia	Muy importante	
Urgencia	Urgente	

RF-04	Mostrar todas las tareas pendientes.	
Descripcion	Se muestra la lista de tareas pendientes.	
Precondicin	Debe existir al menos una tarea pendiente.	
Secuencia	Paso	Accin
	1	Seleccionar la opcin de ver las tareas pendientes.
Postcondicin	El sistema mostrar todas las tareas pendientes	
Excepciones	Paso	Accin
	1	No hay tareas pendientes para mostrar.
Rendimiento	Paso	Cota de tiempo
	1	1 segundo
Importancia	Vital	
Urgencia	Urgente?	

RF-05	Mostrar el horario general del usuario.	
Descripcin	Se muestra el horario completo del estudiante.	
Precondicin	El usuario debe haber creado un horario antes.	
Secuencia	Paso	Accin
	1	Seleccionar la opcin de mostrar el horario.
Postcondicin	El sistema mostrar el horario con las materias, los descansos, los horario de transporte y las tareas pendientes	
Excepciones	Paso	Accin
	1	No hay un horario para presentar.
Rendimiento	Paso	Cota de tiempo
	1	1 segundo
Importancia	Importante	
Urgencia	Puede esperar	

RF-14	Modificar horarios	
Descripcion	Se selecciona y modifica una franja del horario.	
Precondicin	El horario que lo que se desea modificar debe estar asignado.	
Secuencia	Paso	Accin
	1	Se selecciona la opcin de modificar horario.
	2	Se selecciona el horario de la tarea que se desea cambiar.
	3	Se selecciona la nueva franja de horario en la que se acomodara la tarea.
	4	El cambio de horario se ha realizado.
Postcondicin	El horario es modificado y el sistema puede ofrecer sugerencia de tiempo de realizacin, o incluso si se debe sacrificar algn espacio de descanso.	
Excepciones	Paso	Accin
	1	No hay ningn horario para seleccionar, en este caso el caso de uso acaba.
	3	No existe ninguna franja disponible para cambiar, en este caso el caso de uso acaba.
Rendimiento	Paso	Cota de tiempo
	1	1 segundo
	2	1 segundo
	3	10 segundos
	4	1 segundo
Importancia	Importante	
Urgencia	Hay presin	

RF-15	Modificar Tareas.	
Descripcion	Se permite modificar los diferentes campos de una tarea.	
Precondicin	Debe existir alguna tarea para modificar.	
Secuencia	Paso	Accin
	1	Se solicita modificar una tarea.
	2	El usuario selecciona la tarea que desea modificar.
	3	El usuario selecciona el campo de la tarea que desea modificar.
	4	Se modifica el campo de la tarea.
	5	Se permite elegir realizar otro cambio o terminar.
Postcondicin	La tarea tiene un campo modificado, segn el tipo podra haber una sugerencia, como en el caso de dificultad, o cambio de horario para realizarse.	
Excepciones	Paso	Accin
	3	Si la tarea solo tiene los campos minimos se puede agregar el cambio, as el caso de uso continua.
Rendimiento	Paso	Cota de tiempo
	1	1 segundo
	2	1 segundo
	3	1 segundo
	4	5 segundos
	5	1 segundo
Importancia	vital	
Urgencia	Hay presin	

RF-16	Sugerir horarios para realizar tareas.	
Descripcion	Segn los horarios disponibles, al momentos de adicionar una tarea se hace una sugerencia de horario para realizarla.	
Precondicin	Se debe haber agregado una tarea, y deben haber horarios disponibles para hacer la recomendacin.	
Secuencia	Paso	Accin
	1	Al momento de agregar una tarea, se revisan horarios disponibles.
	2	Se revisan las variables de la tarea (dificultad, fecha de entrega).
	3	Se hace la recomendacin.
Postcondicin	La recomendacin se dar al usuario dandole la posibilidad de tomarla o dejarla.	
Excepciones	Paso	Accin
	1	Puede pasar que no haya horarios disponibles para hacer la recomendacin, as el caso de uso termina.
	2	Si no hay variables de tarea definidos, se pasa al siguiente paso.
Rendimiento	Paso	Cota de tiempo
	1	5 segundos
	2	5 segundos
	2	1 segundo
Importancia	Normal	
Urgencia	Puede esperar	
Comentarios	No.	Descripcin
	1	El caso de uso esta bastante ligado a otros casos de uso, como puede apreciarse en el diagrama, sin embargo su relevancia no es la misma como la de los casos a los que apoya.

RF-19	Alertar de la prxima entrega de una tarea.	
Descripcion	Se pretende avisar con tiempo prudencial que el ciclo de una tarea esta por finalizar, lo cual significa que debe ser terminada para ser entregada.	
Precondicin	La existencia de la tarea.	
Secuencia	Paso	Accin
	1	Se revisa el tiempo para que la tarea deba ser terminada comparandolo con el prudencial.
	2	Se realiza el aviso.
Postcondicin	El usuario ser avisado sobre la proximidad de su tarea.	
Excepciones	Paso	Accin
	1	Si la tarea no cuenta con tiempo de realizacin se considera indefinida, as el caso de uso termina.
Rendimiento	Paso	Cota de tiempo
	1	1 segundo
	2	1 segundo
Importancia	Importante	
Urgencia	Puede esperar	

Capítulo 4

Interaccin

4.1. Introduccin

cntenido...

Capítulo 5

Clases

5.1. Introduccin

Los diagramas de clase son parte importante del diseo de un software, ya que estos permiten generar diseos que plasmen la solucin a un problema, la cual ser entendible para todos aquellos conozan del lenguaje unificado de modelado (UML).
Adems es necesario utilizar diagramas de clase para representar grficamente y de forma esttica la estructura general del sistema, mostrando sus clases e interacciones.

5.2. Teora

Los diagramas de clase sirven para visualizar las relaciones entre las clases que involucran el sistema,entre estas se encuentran:

- Dependencia.
- Asociacin.
- Agregacin.
- Composicin.
- Generalizacin.
- Realizacin.

Estas relaciones pueden subdividirse en dos grandes grupos: Las relaciones cliente/proveedor, en las cuales entran las dependencias y asociaciones

(asociacin, agregacin y composicin), las cuales generan un alto acoplamiento en el software, pero tambin lo hacen seguro. Por otro lado estn las relaciones de generalizacion en las cuales estn la especializacin e implementacion. Estas poseen un bajo acoplamiento, pero no poseen la seguridad de las de cliente/proveedor. El ideal es crear un diagrama de clases en el que haya un equilibrio entre el acoplamiento y la seguridad.

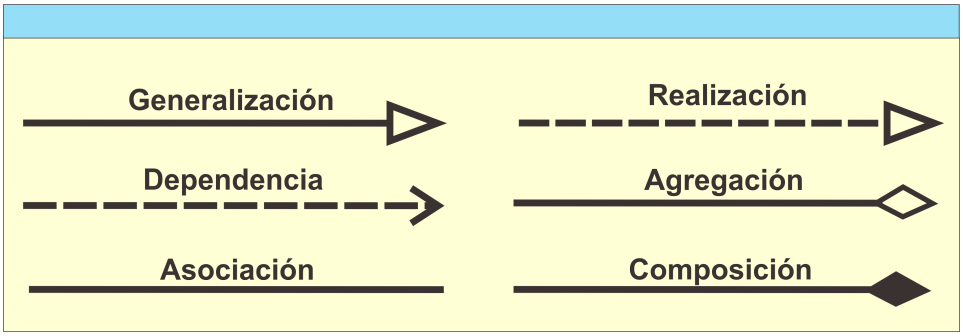

Figura 5.1: Relaciones UML. Tomada de internet

Por otro lado, las clases se representan por medio de un rectngulo que se divide en tres:

- Nombre de la clase: Cmo se denomina la clase. Es un sustantivo.
- Atributos de la clase: Pueden ser de diferentes tipos (booleano, numrico, cadenas Y T.D.A), tienen modificadores de visibilidad (public, private, protected, de paquete), nombre y propiedades (final, const) si es necesario.
- Mtodos de la clase: Son las operaciones que realiza la clase, deben denominarse con verbos, poseen modificadores de visibilidad y pueden retornar diferentes tipos de datos (void, numrico, cadenas y T.D.A) Adicionalmente, los mtodos pueden recibir argumentos, que se representan en las operaciones en forma de parmetros los cuales son variables que poseen nombre y tipo.

Capítulo 6

Patrones

6.1. Introduccin

Aunque histricamente el software pueda considerarse como primitivo si se compara con otras disciplinas que llevan siglos de ventaja en cuanto a su desarrollo, actualmente no es as. La anterior reflexin se deduce debido a que el software ha evolucionado lo suficiente para convertirse en una herramienta que va mas all de solo programar, tambin se encarga de disear y modelar. En este caso, el diseo de software puede generar soluciones generalizadas a problemas reincidentes. As es como llegamos a los patrones de diseo, los cuales para cualquier proyecto, independientemente de su complejidad y tamao, seguramente se vern involucrados.

A partir de lo anterior, es claro que los patrones de diseo aparecern en la propuesta de estructura del proyeto, ms conocida como diagrama de clases, con la intencin de obtener un diagrama razonable desde un punto de vista de principios de diseo as como los principios del paradigma orientado a objetos.

Se debe mencionar tambin que los patrones propuestos en este documento, no necesariamente todos estn incluidos en el Gof (the Gang of Four), pues existen otras soluciones que aunque no tan conocidas, pueden tener un impacto positivo debido en el desarrollo del software.

6.2. Patrn Composite

El patrn de diseo Composite (Componente), nos permite construir estructuras complejas partiendo de otras estructuras mucho mas simples, lo cual permite crear estructuras compuestas conformadas por otras mas pequeas. Este patrn resulta til para la creacin de subtareas dentro de una tarea mas general, ya que se genera una estructura en forma de rbol gracias a la recursividad con la que funciona el patrn. Otra ventaja de su utilizacin, en este desarrollo especficamente, es que se puede representar la jerarqua de tarea-subtarea, adems de aadir dinamismo a la tarea, ya que sta puede tener subtareas de diferentes tipos. Adems es posible tratar la subtarea como tarea.
En conclusin, el patrn componente posibilita la solucin del problema de las subtareas, ya que permite jerarquizar, aadir dinamismo a la tarea por medio de subtareas y construir la tarea general por medio de subtareas, por esta razn ser utilizado dentro de este software.

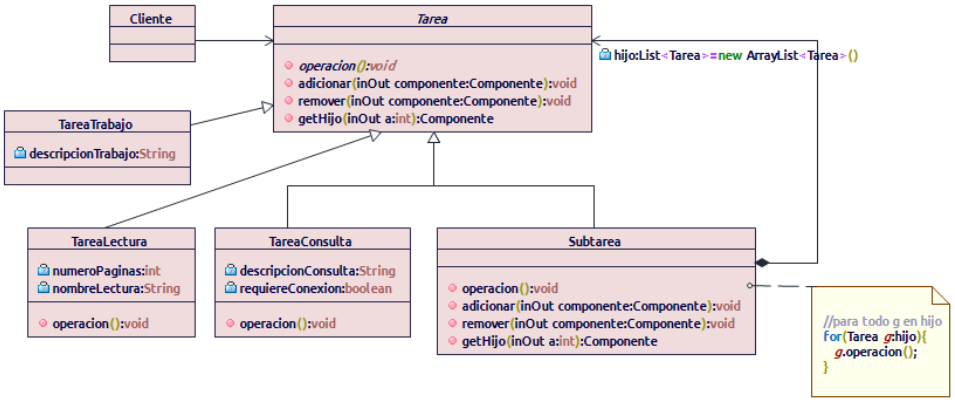

Figura 6.1: Patrn Componente

6.3. Patrn agrupador

Un horario puede mostrarse como la constitucin de diversas franjas en un orden lgico; estas deben agruparse para que haya orden y se puedan manejar conjuntamente. Por est razn el patrn agrupacin ser de utilidad, ya que permite generar una estructura en la cual los mdulos, que en este caso son las franjas, puedan ser agrupados para invocarse de forma colectiva como el horario, de esta forma se centraliza el control de la estructura en una sola clase.

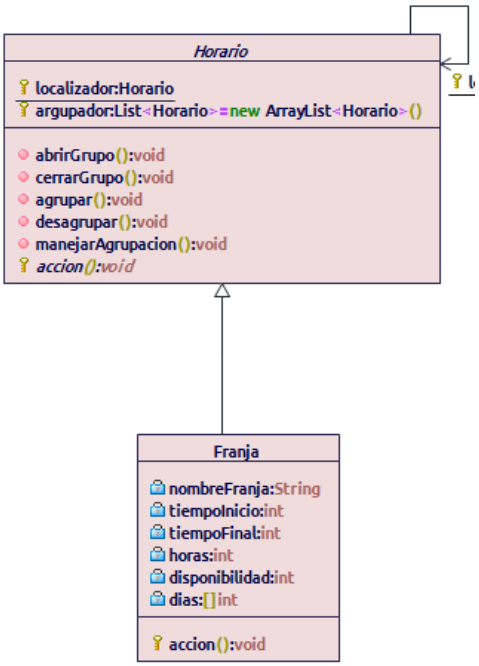

Figura 6.2: Patrn Agrupador

6.4. Patrn Fbrica Abstracta

La creacin de objetos es una situacin que aparece en prcticamente cualquier proyecto, por lo que se debe buscar una estructura que permita llevar esta creacin de la mejor manera, incluso en mayor medida cuando el objeto a crear pertenece a una familia de otros objetos que tambin pueden ser creados, y por supuesto en un caso ms general en caso de que hayan varias de estas familias de objetos.

La fbrica abstracta lo que hace es esto, proveer una interfaz para la creacin de estas familias de objetos relacionados, sin especificar sus clases concretas.

Para el caso del proyecto, la fbrica abstracta es til pues existen dos familias de objetos a crear, una de tareas referente a lo acadmico y otra la de tareas que no tienen que ver con la universidad.

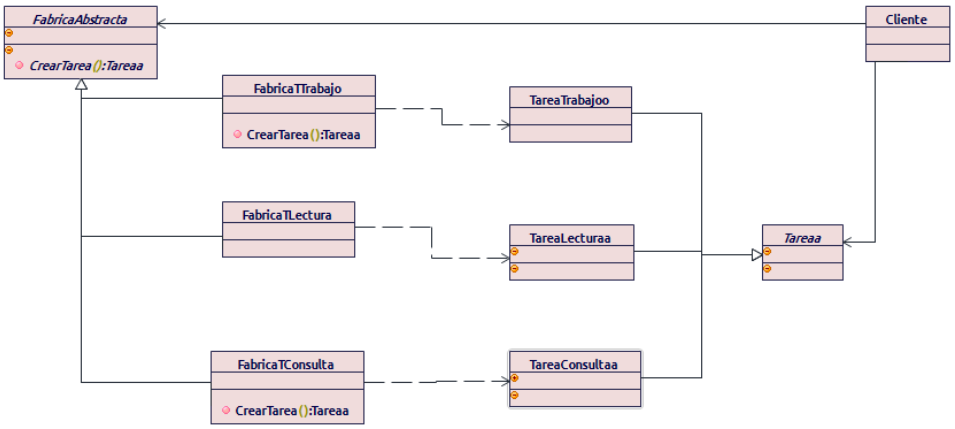

Figura 6.3: Patrn Fbrica Abstracta

6.5. Patrn Estrategia

Dada la situacin de que un programa pueda ofrecer un servicio, el cual se pueda realizar de varias maneras hace alusin a este patrn. La intencin de este es poder seleccionar la alternativa ms adecuada para el cliente, durante tiempo de ejecucin.

En la mencin a la fbrica abstracta se mencionaron las familias de objetos de tarea y categoria, las cuales en algn momento necesitarn de la posibilidad de una modificacin de alguno de sus objetos, operacin que no solo involucra a este ltimo en cuestin, sino que abarca principalmente el modulo externo de manejo de base de datos. Dependiendo de cada objeto, la misma operacin debe hacerce de manera distinta, haciendo as que entre el patrn estrategia.

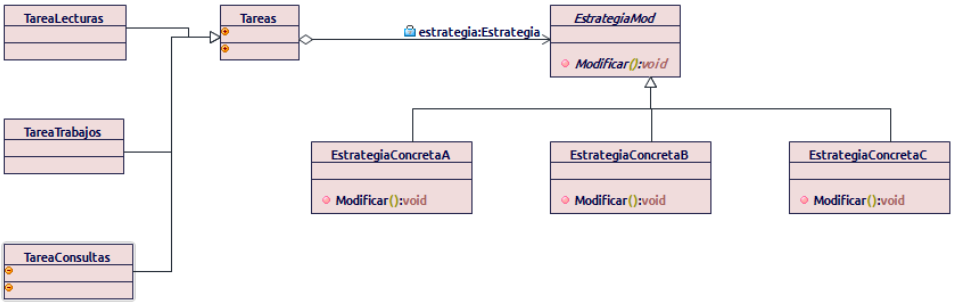

Figura 6.4: Patrn Estrategia

Capítulo 7

Estados

7.1. Introduccin

cntenido...

Capítulo 8

Componentes

8.1. Introduccin

cntenido...

Capítulo 9

Nodos

9.1. Introduccin

cntenido...

Capítulo 10

Actividades

10.1. Introduccin

cntenido...

Parte III

REFLEXIONES

Capítulo 11

Conclusiones

11.1. Introduccin

cntenido...

Apéndice A

Apndice captulo 6: Patrones

A.1. Patrn composite

A.1.1. Clase Tarea

```
1  public abstract class Tarea{
2  public abstract void operacion();
3  public void adicionar(Componente componente){}
4  public void remover(Componente componente){}
5  public Componente getHijo(int a){
6    return null;
7  }
8  }
```

A.1.2. Clase Subtarea

```
9  import java.util.List;
10 import java.util.ArrayList;
11 public class Subtarea extends Tarea{
12 private List<Tarea> hijo=new ArrayList<Tarea>();
13 public void operacion(){
14   //para todo g en hijo
15   for(Tarea g:hijo){
16      g.operacion();
17   }
18 }
19 public void adicionar(Componente componente){
20   hijo.add(componente);
21 }
22 public void remover(Componente componente){
23   hijo.remove(componente);
```

```
24 }
25 public Componente getHijo(int a){
26   return hijo.get(a);
27 }
28 }
```

A.1.3. Clase Tarea trabajo

```
29 public class TareaTrabajo extends Tarea{
30 private String descripcionTrabajo;
31 }
```

A.1.4. Clase Tarea consulta

```
32 public class TareaConsulta extends Tarea{
33 private String descripcionConsulta;
34 private boolean requiereConexion;
35 public void operacion(){}
36 }
```

A.1.5. Clase Tarea lectura

```
37 public class TareaLectura extends Tarea{
38 private int numeroPaginas;
39 private String nombreLectura;
40 public void operacion(){null}
41 }
```

A.2. Patrn agrupador

A.2.1. Clase horario

```
42 import java.util.List;
43 import java.util.ArrayList;
44 public abstract class Horario{
45     protected static Horario localizador;
46     protected List<Horario> argupador=new ArrayList<Horario
       >();
47     public static void main(String[] args){
48         Horario A =  new Franja();
49         A.abrirGrupo();
50         A.agrupar();
51         A.manejarAgrupacion();
52         Horario B =  new AgrupadorB();
53         B.agrupar();
54         B.manejarAgrupacion();
```

```
55      }
56      public void abrirGrupo(){
57          if(localizador==null){
58              localizador=this;
59              argupador=new ArrayList<Horario>();
60          }
61      }
62      public void cerrarGrupo(){
63          localizador = null;
64      }
65      public void agrupar(){
66          if(localizador!=null){
67              (argupador = localizador.argupador).add(this);
68          }
69      }
70      public void desagrupar(){
71          if(localizador!=null){
72              localizador.argupador.remove(this);
73          }
74      }
75      public void manejarAgrupacion(){
76          for(Horario a:argupador){
77              a.accion();
78          }
79      }
80      protected abstract void accion();
81  }
```

A.2.2. Clase franja

```
82  import static com.componentes.diseno.lmc.marcosDeReferencia
        .computacion.Computacion.*;
83  import static com.componentes.diseno.lmc.marcosDeReferencia
        .emoticons.Emoticons.*;
84  public class Franja extends Horario{
85      private String nombreFranja;
86      private int tiempoInicio;
87      private int tiempoFinal;
88      private int horas;
89      private int disponibilidad;
90      private []int dias;
91      protected void accion(){
92          System.out.println(SERIOUS);
93      }
94  }
```

A.3. Patrn fbrica abstracta

A.4. Patrn estrategia

Bibliografía