e chapterReferencias bibliográficas42table.Alph1.1

Título del Trabajo Fin de Grado

Nombre del Alumno

Trabajo fin de Grado

Supervisado por Dr. Pablo Trinidad Martín-Arroyo

Universidad de Sevilla

enero 2026

Publicado en enero 2026 por
Nombre del Alumno

http://www.lsi.us.es/~trinidad
ptrinidad@us.es

Pon aquí cuestiones acerca del copyright

Yo, D. Nombre del Alumno con NIF número 12345678A,

DECLARO

mi autoría del trabajo que se presenta en la memoria de este trabajo fin de grado que tiene por título:

Título del Trabajo Fin de grado

Lo cual firmo,

Fdo. D. Nombre del Alumno
en la Universidad de Sevilla
28/01/2026

Tu dedicatoria aquí

Agradecimientos

No olvides añadir una nota de agradecimiento a quienes hayan contribuido emocionalmente al proyecto fin de Grado.

Resumen

Un resumen de un párrafo sobre el problema planteado en el proyecto y la solución. Máximo 300 palabras.

Índice general

Índice de figuras

Índice de cuadros

Contexto

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

Ernest Hemingway (1899–1961),
Novelist

Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?

1.1 El mundo del X (videojuego, e-commerce,...)

Hay que ir poco a poco acotando el contexto donde se desarrolla el proyecto. No se debe sobreentender que el evaluador de la memoria sabe del tema. Escribid el texto para la abuela.

1.2 Subcontexto

1.3 Subsubcontexto

1.4 Estado del arte

Cómo se encuentra la industria hoy en día a nivel económico y tecnológico.

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

Ernest Hemingway (1899–1961),
Novelist

Aquí mal un breve resumen del capítulo.

2.1 Motivación

Esta sección se rellenará cuando tengamos un producto de mercado en lugar de un proyecto en el que haya un cliente específico. Deberá justificar brevemente el problema a resolver, escenario en el que se aplica, hipótesis de partida, público objetivo, etc.

2.2 Listado de objetivos

Objetivo 1. Blabla Detalles del objetivo 1.

Objetivo 2. Blabla Detalles del objetivo 2.

Metodología

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

Ernest Hemingway (1899–1961),
Novelist

Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?

3.1 Estructura organizacional del proyecto

¿Se hace en grupo? En caso afirmativo, ¿cuál va a ser la responsabilidad de cada uno?

3.2 Metodología de desarrollo

Indicar en qué metodología nos basamos, explicarla brevemente y luego adaptarla a nuestras necesidades. Cada una de estas cuestiones debe ser una subsección.

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

Ernest Hemingway (1899–1961),
Novelist

Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?

4.1 Resumen temporal del proyecto

Resumen del proyecto	
Fecha de inicio	10/10/2014
Fecha de fin	10/10/2014
Periodicidad de las revisiones	3 semanas
Carga de trabajo semanal	12 horas
Horas totales previstas	225 horas
Horas finales	234 horas

Cuadro 4.1: Tabla resumen de tiempos y planificación

4.2 Planificación inicial

Aquí un desglose de las iteraciones, comienzo y fin de cada una:

Resumen de iteraciones	
Iteración 1	10/10/14 a 21/10/14
Iteración 2	21/10/14 a 15/11/14
...	dd/mm/aa a dd/mm/aa

Cuadro 4.2: Planificación temporal de iteraciones

Explicar cómo se han decidido las fechas, interacción con fechas importantes y situaciones personales.

ESTE CAPÍTULO DEBE ESCRIBIRSE AL COMIENZO DEL PROYECTO

4.3 Informe de tiempos del proyecto

Lo mismo que el anterior pero con datos reales. Ver Tabla §4.3.

Justificar los retrasos de forma detallada aquí para cada una de las iteraciones. Explicar las razones.

Resumen de iteraciones	
Iteración 1	10/10/14 a 21/10/14
Iteración 2	21/10/14 a 15/11/14
...	dd/mm/aa a dd/mm/aa

Cuadro 4.3: Planificación temporal de iteraciones

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

Ernest Hemingway (1899–1961),
Novelist

Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?

5.1 Resumen de costes del proyecto

Resumen del proyecto	
Costes de personal	5.045 €
Sueldo neto	2.030 €
Impuestos	1.000 €
Costes sociales	2.015 €
Costes materiales	560 €
Costes indirectos	450 €
TOTAL	8.000 €

Cuadro 5.1: Tabla resumen de costes

5.2 Costes de personal

Ya hablaremos de esto

5.3 Costes materiales

Y de esto también. Ver Sección §6.2.

5.4 Costes indirectos

Y esto es una fiesta

Arranque

Este capítulo define la arquitectura base del proyecto Scubex y la lista inicial de características priorizadas para el desarrollo del Producto Mínimo Viable (MVP), centrado en la exploración de biodiversidad marina.

6.1 Lista de características

Siguiendo la metodología de desarrollo, se han identificado las siguientes características nucleares:

1. **Gestión de Usuarios (OAuth2)**: Autenticación delegada mediante Google.
2. **Exploración Geográfica**: Visualización de mapa interactivo con datos climáticos.
3. **Escáner de Especies**: Identificación de fauna marina en tiempo real basada en coordenadas.
4. **Perfil de Usuario**: Persistencia de preferencias y datos básicos.

6.2 Diseño arquitectónico

El sistema sigue una arquitectura de cliente-servidor desacoplada.

6.2.1 Stack Tecnológico

- **Backend**: Spring Boot 4 (Java 21). Actúa como Gateway de APIs científicas.
- **Frontend**: React 18 + Vite. SPA para una experiencia de usuario fluida.
- **Base de Datos**: H2 (Embebida) para datos de usuario; sin persistencia propia de datos biológicos (se consumen en tiempo real).
- **Seguridad**: Spring Security (OAuth2 Client).

6.2.2 Integración de APIs Externas

- **OBIS**: Proveedor de ocurrencias biológicas (Nombre científico, Coordenadas, Taxonomía).
- **iNaturalist**: Proveedor de metadatos multimedia (Fotos, Nombres comunes).
- **WeatherAPI**: Datos meteorológicos (Viento).
- **Stormglass**: Datos oceanográficos (Temp. agua) [Uso restringido por rate-limit].

Esta iteración aborda el núcleo funcional de Scubex: la capacidad de identificar especies marinas en una zona determinada. Se ha seguido un ciclo TDD para implementar la compleja lógica de integración y filtrado de datos entre múltiples APIs científicas.

7.1 Características a desarrollar

1. Escáner de Especies bajo demanda (Backend). Ver Tabla §7.1.

2. Filtrado y Enriquecimiento de Datos Biológicos.

Análisis de valor aportado: Escáner de Especies	
Propuesta	Implementación de un endpoint REST que agregue datos de OBIS e iNaturalist para una zona geográfica.
Valor	Permite al usuario descubrir fauna real validada científicamente sin conocimientos técnicos.
Coste	Alto esfuerzo de integración. Requiere orquestar llamadas asíncronas y gestionar distintos formatos de datos.
Opciones	Usar solo una API (OBIS) reduciría el coste, pero eliminaría el valor visual (fotos) y semántico (nombres comunes), haciendo la app inútil para turistas.
Riesgos	Rate limits de APIs externas. Inconsistencia de datos entre OBIS e iNaturalist.
Deuda técnica	Complejidad en la gestión de errores distribuidos (si una API falla).

Cuadro 7.1: Análisis de valor aportado: Escáner

7.2 Diseño

El diseño se centra en resolver la incompatibilidad entre la búsqueda radial del usuario y la búsqueda geométrica requerida por las fuentes de datos.

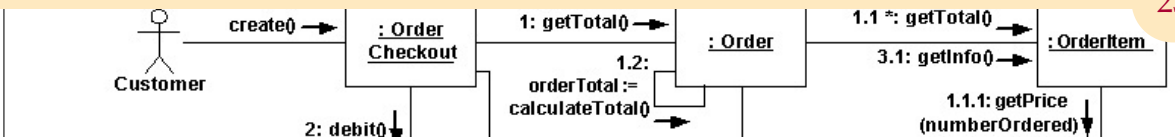
7.3 Implementación

Se ha aplicado TDD para definir primero el comportamiento del orquestador de especies. A continuación se detalla el flujo principal.

Memorando técnico 001: Geometría de Búsqueda	
Asunto	Incompatibilidad de búsqueda radial en API OBIS.
Resumen	Generación de polígonos WKT a partir de radio y centro.
Factores causantes	El frontend solicita datos basados en un punto central y un nivel de zoom (radio), pero OBIS requiere un objeto 'POLYGON' en formato WKT.
Solución	Implementar un servicio de utilidad geométrica que calcule los vértices de un cuadrado (o aproximación circular) inscrito en el radio de visión.
Motivación	OBIS ofrece la mayor base de datos científica, es imperativo adaptarse a su interfaz.
Alternativas	Filtrado post-proceso: Pedir un área muy grande y filtrar en memoria (descartado por ineficiencia).

Cuadro 7.2: Memorando técnico 001: Geometría

Identificador	Descripción de la acción de alto nivel		
SCAN-01	Orquestación de Escaneo		
Métodos de alto nivel			
[List<Species>] scanSpecies (lat: Double, lng: Double, zoom: Integer)			
Pasos (Usar Pseudocódigo o similar)			
1. Calcular radio de búsqueda basado en el nivel de zoom.			
2. Generar polígono WKT (Well-Known Text) para el área definida.			
3. Consultar OBIS API con el polígono para obtener ocurrencias raw.			
4. Para cada especie única devuelta por OBIS:			
4.1. Consultar iNaturalist API por nombre científico.			
4.2. Si iNaturalist devuelve 0 resultados, descartar especie.			
4.3. Si hay resultados, enriquecer con Foto y Nombre Común.			
5. Retornar lista filtrada y ordenada.			
Métodos de bajo nivel necesarios			
Paso	**Clase**	**Método**	**Mem. Técn.**	**IU**
1	ZoomUtils	[Double] calculateRadius(zoom)	001	NO
2	GeometryUtil	[String] createPolygonWKT(lat, lng, radius)	001	NO
3	ObisClient	[List<ObisResult>] fetchOccurrences(wktPolygon)	001	NO
4.1	INatClient	[INatTaxon] searchTaxon(scientificName)	002	NO
4.2	Stream	filter(taxon -> taxon.getTotalResults > 0)	002	NO
Diagrama de Colaboración				



Memorando técnico 002: Calidad de Datos	
Asunto	Presencia de microorganismos no visualizables.
Resumen	Filtrado cruzado mediante existencia de recursos en iNaturalist.
Factores causantes	OBIS devuelve todo tipo de registros biológicos (ej. *Pycnococcaceae*). Para una app turística/deportiva, estos datos son ruido.
Solución	Verificar cada nombre científico contra iNaturalist. Si 'total_results == 0' o no hay foto, se descarta el registro.
Motivación	Mantener la relevancia visual y la experiencia de usuario.
Cuestiones abiertas	El aumento de latencia por múltiples peticiones HTTP.

Cuadro 7.3: Memorando técnico 002: Filtrado

7.4 Pruebas

Siguiendo TDD, se implementaron los siguientes casos antes del código productivo:

- **shouldGenerateValidWKT**: Verifica que, dado un punto y radio, se genera un String 'POLYGON(...)' geométricamente cerrado.
- **shouldFilterMicroorganisms**: Mockea una respuesta de OBIS con *Pycnococcaceae* y verifica que, al simular una respuesta vacía de iNaturalist, el servicio retorna una lista vacía.

7.5 Despliegue

El backend se ha empaquetado como un JAR ejecutable (Spring Boot) y se despliega junto con la configuración de credenciales para las APIs externas.

Manual de usuario

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

Ernest Hemingway (1899–1961),
Novelist

Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?

8.1 Sección libre

Estructurar en función del proyecto.

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

Ernest Hemingway (1899–1961),
Novelist

Resumen de lo que va a ocurrir en el capítulo. ¿Cuál es el objetivo que tenemos con este capítulo?

9.1 Informe post-mortem

Qué es un informe post-mortem

9.1.1 Lo que ha ido bien

- Argumento a favor 1.
- Argumento a favor 2.
- Argumento a favor 3.

9.1.2 Lo que ha ido mal

- Argumento en contra 1.
- Argumento en contra 2.
- Argumento en contra 3.

9.1.3 Discusión

En función de lo anterior, qué cambiaría si empezara hoy el proyecto de nuevo.

9.2 Trabajos futuros

Enumera los puntos abiertos y que no se han resuelto. Indica si darían lugar a otro proyecto y de qué forma se podría acotar.

Parte V

Appendices

The good parts of a book may be only something a writer is lucky enough to overhear or it may be the wreck of his whole damn life – and one is as good as the other.

Ernest Hemingway (1899–1961),
Novelist

This is an example of an abstract. Multiple lines are supported. Several paragraphs. It jumps to the next page. Blau blau blau. I am introducing more text to reach the third line

A.1 Software Product Lines

- Objective of a Product Line (PL) (mass production and customisation) [1]
- The focus in software derives in Software Product Lines (SPLs).
- Variability management: variability models
- When and how are used VMs: FMs are described in FODA report as a key element in SPL since they represent the variability and commonality of the different products in a SPL.

A.2 Feature Models

To Abductive Section in 2.1

As the number of products to be built by a SPL may be large and the constraints among features may be complex, representing such an information in a manageable and compact manner is a must. Feature Models (FMs) represent the set of products a SPL may build in terms of product features. Some features are optional while others are mandatory. To indicate the relationships among features, they are hierarchically linked, forming a tree whose root is a feature representing the whole functionality of a product. The root feature is refined in child features, which increase the level of detail and reduce the scope of features. Recursively following this refinement process, a tree-like structure is obtained where three basic kinds of hierarchical relationships are used:

- Mandatory: a mandatory relationship affects a parent and child feature. It forces the child feature to appear in a product whenever its parent feature does.
- Optional: a child feature connected to a parent feature by means of an optional relationship may be optionally selected whenever its parent feature is.
- Set-relationships: three or more features are part of a set-relationship: a parent feature and a set of two or more child features. A set-relationship contains a cardinality that constraints the number of child features to be selected in a product whenever its parent feature is selected. If the cardinality is $[1.,1]$ it is commonly remarked as an *alternative relationship* where only one child feature may be selected at the same time. If the cardinality is $[1..N]$ (where N is the number of

child features), it is also known as an *or-relationship* as any combination of child features is allowed while at least one is selected.

Although FMs can represent most of the most frequent constraints, the hierarchical nature of these models might hinder the representation of some constraints. Under this circumstance, *cross-tree constraints* can be added. The most common kinds of cross-tree constraints are:

- Dependency: a feature depends on another feature if the second one must be part of a product whenever first one is selected.
- Exclusion: two features exclude themselves if both of them cannot be part of a product at the same time.

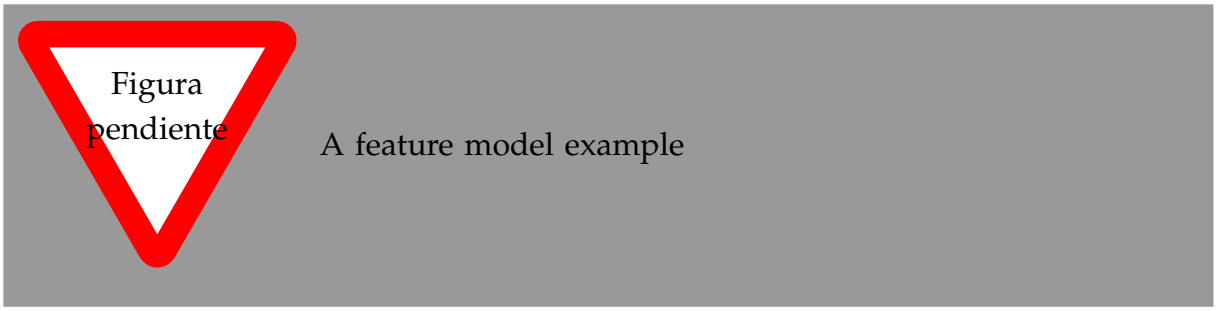


Figura A.1: An example of a Home Integration System

The example in Figure §A.1 describes a *Home Integration System* (HIS) SPL in terms of its features and the relationships among them. Leaning on this example we define some useful terms:

Partial configuration : a partial configuration is a composed by three sets of selected (S), removed(R) and undecided(U) features. A feature can only be in one of these sets and every feature in the FM (fm) must be in one of them, i.e. $S \cup R \cup U = fm$ and $S \cap R \cap U = \emptyset$. A partial configuration represents an intermediate state during the process of a customer selecting the feature for a custom product. For example, $S_P = \{...\}$, $R_P = \{...\}$ and $U_P = \{...\}$ define a partial configuration for the sample FM where some features are still to be decided if they are to be selecter or removed in a configuration.

(Full) configuration : a full configuration or simply a configuration is a partial configuration such that the set of undecided features in empty. For example, $S_F = \{...\}$ and $R_F = \{...\}$ describe a full configuration for the example FM.

Product : a product is a representation for a full configuration such that only the selected features are remarked. For instance, $P = \{\}$ is a product for the above full configuration. A product such as `A,B` is a valid since all the constraints within the FM are satisfied. However, A,B and C is not a valid product since D is required.

Validation A partial configuration is *valid* if all the relationships and constraints are satisfied given the sets of selected, removed and undecided features. So the definition applies for valid full configurations and valid products. As a conclusion we can affirm that a FM represents all the valid products in a SPL.

Objetivo: Briefly expose attributes as an important asset in feature models.

It is frequent that features are not enough to represent information that is relevant to represent a SPL variability. In this case, FMs are extended with feature attributes such as cost, versions, RAM consumption, etc. in the so-called Extended Feature Models (EFMs) [1]. Besides relationships, an EFM contains constraints that affect attributes which reduce even more the set of products a FM describes. Above definitions remain when attributes are introduced into FMs.

A.3 Automated Analysis of Feature Models

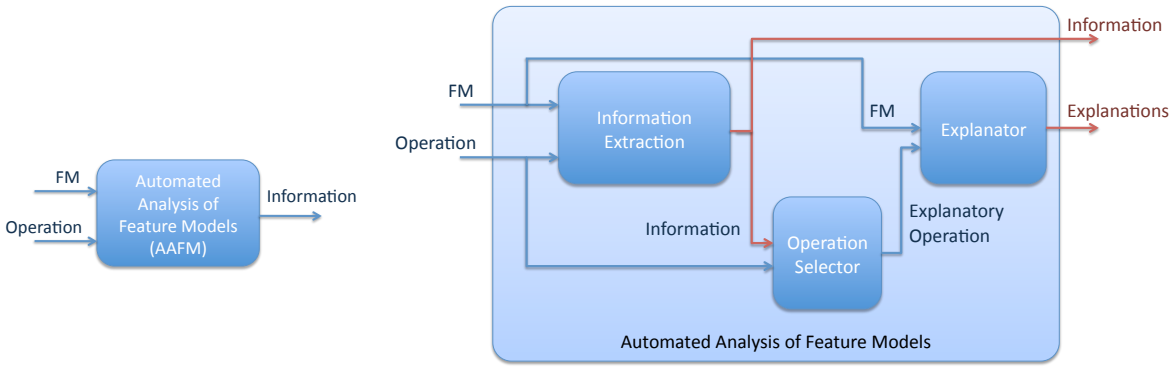
A.3.1 Scope

To Abductive Intro

FMs are used all along the SPL development as key models and many of the development decisions are taken relying on the information contained within them. Most of the times, relationships are complex and hinder the manual extraction of information. Manually obtaining information such as 'which is the product that costs the less?', 'does the feature model contain errors?' or 'why there exist no product containing certain features?' can be an unfeasible task. The complexity and compactness of FMs justify the need of an automated support of these operations. So the *Automated Analysis of Feature Models* (AAFM) arises as a topic of interest to deal with this problem in the SPL community.

The AAFM can be seen as a black-box process that receives a FM and an operation as inputs and obtains information (its kind depends on the analysis operation) as an

output (Fig. A.2(a)). There are many operations that extract information from a FM such as 'counting products' operation whose result is a natural number indicating the number of customised products that can be built; or 'list of products' operation that obtains each of those products. This vision of AAFM as a black-box is valid for a subset of analysis operations that we call *information extraction operations* (IEO) that can be seen as processes to extract information from FMs. In other words, an IEO makes explicit an implicit information within a FM.



(a) The AAFM seen as a black-box process

(b) Extending the AAFM process with explanations

Figura A.2: A different view on AAFM distinguishing between information extraction and explanatory operations

Use me to explain in a larger text than 'side-text' anything that is important to a reader not familiar with the dissertation context for example.

However, there is a subset of analysis operations known as *explanatory operations* (EO) whose objective is explaining the result obtained from a IEO. Sometimes, the result is not the expected one and the analyser needs to know which are the relationships that have caused it. For example, let us suppose that the IEO 'which are the products described in a FM that cost less than $1000?' obtains no products as a result. If we were expecting to obtain at least one product, it is important to determine the relationships in the FM that are responsible of that behaviour, so an EO 'why there is no product costing less than $1000?' will shed light on the relationships that avoid obtaining any product. Obtaining no result is not the only case that claims for explanations. If we obtained only one product as a result and we were expecting to obtain at least 10 products, although an answer is obtained the result is unexpected and the discrepancy reasons have to be found. Moreover, explanatory operations are also use-

ful even when an expected result is obtained, to reinforce the certainty that the result is correct. So it can be concluded that EOs complement the information an FM analyser obtains from IEOs.

The complexity of feature modelling relies on correctly setting the relationships that describe the set of products to be built in a SPL. Relationships are the only elements responsible of the results obtained in FM analysis. So an *explanation* is a set of relationships that may have caused that result. While IEO provides for an unique response that is known for certain, an EO provides for a set of probable explanations to a result obtained from a IEO, being only one of them a valid explanation. It would be the analyser the one in charge of discriminating the correct explanation, maybe performing new analysis operations.

THIS IS A SIDE TEXT. USE TO REMARK IMPORTANT INFORMATION

Therefore, two kinds of operations are distinguished in AAFM: information extraction and explanatory operations. Explanatory operations have no sense without a paired information extraction operation and its result. To ensure that explanatory operations are always paired to an information extraction operation, we define a new black-box process of AAFM that incorporates explanations as an additional output (see Figure A.2(b))

1. Information extraction: the original process, which remains the same.

2. Operation selector: depending on the information extraction operation the analyser asks for and the information obtained as a result, this process provides the explanatory operation to be performed. In other words, it pairs an explanatory operation to an information extraction operation.

3. Explanatory analysis: provides a set of explanations from the FM and the explanatory operation.

The overall process can be encapsulated into a holistic black-box process which receives the FM and the information extraction operation as inputs and provides a result and explanations as outputs. It can be seen as we just add explanations as an output to the analysis process.

To realise this view on the AAFM, we need to give details on the insides of these black-boxes. Since the information extraction process is already rigourously defined in

Benavides' PhD dissertation, the purpose of this paper is defining the remaining two sub-processes. We formalise the explanatory analysis process by means of default logic and provide the criteria to implement the operation selector process.

Most Common Techniques to perform AAFM Operations.

A.4 Dynamic Software Product Lines (DSPL)

What is a Dynamic Software Product Line (DSPL). Different points of view. What is important is the automation of reconfiguration properties relying on SPL techniques.

We focus in the application of explanations in DSPLs as an application of our results. Specifically we have worked in MAS and smart homes providing a solution for automating product reconfiguration.

A.5 Hypothesis and Objectives

Objetivo: Justifying that explanations are a particular set of operations in AAFM that are not solvable by means of the techniques that are used up-to-date

Objetivo: Set an impacting phrase that summarises the hypothesis

Hypothesis

Explanations cannot be solved by AI techniques used to solve AAFM. There should exist other AI techniques to solve explanations.

Objective of the dissertation

Defining a framework to provide solutions for explanatory analysis in FMs.

This dissertation summarises our contribution to solve some of the objectives we set in our PhD project.

- Defining a catalog of analysis operations where explanations are applied.
- Rigorously defining these operations in terms of logics.
- Proposing solutions to these operations.
- Validating our results by means of tools and projects where they are applied.

Next chapter focuses on refining how we have contributed to deal with the above objectives.

A piece of code...

```
public Map<Cardinality,CardinalValue> detectWrongCardinals() {
        // any other implementation of Map can be used instead.
        Map<Cardinality,CardinalValue> result =
                new TreeMap<Cardinality,CardinalValue>();
        for( r : relationships) {
                if (r instanceof Set) {
                        Set set = (Set)r;
                        Cardinality card = set.getCardinality();
                        Domain dom = card.getDomain();
                        for (value: dom.getValues())
                                if (isWrongCardinal(card,value))
                                        result.put(card,value);
                }
        }
        return result;
}
```

A coolTable. Use inside a table.

Use \TableSubtitle{n,title} to add a subtitle as the header. n is the number of columns and title is the text to place. [1]

A Catalog of FM Explanatory Operations (2009 version)		
Information Extraction Operation	**FM Explanatory Operations**	
	Why? operation	*Why not? operation*
Valid FM	-	invalid FM
Valid Configuration	valid partial conf.	invalid partial conf.
Valid Product	valid product	invalid product
Products Listing	vaild Product/Config	invalid FM/Product/Config
Products Counting	vaild Product/Config	invalid FM/Product/Config
Optimisation	vaild Product/Config	invalid FM/Product/Config
Core feature	core feature	core feature
Variant feature	variant feature	variant feature
Dead feature detection	-	dead feature
False-optional feature detection	-	false-optional feature
Wrong-cardinality detection	-	wrong cardinal
Information Extraction Operation	**Configuration Explanatory Operations**	
	Why? operation	*Why not? operation*
Valid Configuration	valid partial conf.	invalid partial conf.

Cuadro A.1: Most frequently used explanatory operations and their corresponding information extraction operations

BIBLIOGRAFÍA

[1] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated reasoning on feature models. *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, 3520:491–503, 2005. ISSN 0302-9743. No citation