

ROGUE CARDS

NO SUBTITLE

PEDRO PARRILLA BASCÓN

Trabajo fin de Grado

Supervisado por Dr. Pablo Trinidad Martín-Arroyo



Universidad de Sevilla

julio 2024

Publicado en julio 2024 por

Pedro Parrilla Bascón

Copyright © MMXXIV

<http://www.lsi.us.es/~trinidad>

pparrilla10@gmail.com | pedparbas@alum.us.es

Pon aquí cuestiones acerca del copyright

Yo, D. Pedro Parrilla Bascón con NIF número 77927669N,

DECLARO

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

Rogue Cards,
No subtitle

Lo cual firmo,

Fdo. D. Pedro Parrilla Bascón
en la Universidad de Sevilla
14/07/2024

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

I	Introducción	1
1.	Contexto	3
1.1.	El mundo del Videojuego	4
1.2.	Subcontexto	4
1.3.	Subsubcontexto	4
1.4.	Estado del arte	4
2.	Contexto	5
2.1.	Motivación	6
2.2.	Listado de objetivos	6
II	Organización del proyecto	7
3.	Metodología	9
3.1.	Estructura organizacional del proyecto	10
3.2.	Metodología de desarrollo	10
3.3.	Descripción detallada	10
4.	Planificación	13
4.1.	Resumen temporal del proyecto	14

4.2. Planificación inicial	14
4.3. Informe de tiempos del proyecto	14
5. Costes	17
5.1. Resumen de costes del proyecto	18
5.2. Costes de personal	18
5.3. Costes materiales	18
5.4. Costes indirectos	18
III Desarrollo del proyecto	19
6. Arranque	21
6.1. Lista de características	22
6.2. Diseño arquitectónico	22
7. Costes	23
7.1. Características a desarrollar	24
7.2. Diseño	24
7.3. Implementación	24
7.4. Pruebas	26
7.5. Despliegue	26
IV Cierre del proyecto	27
8. Manual de usuario	29
8.1. Sección libre	30

9. Conclusiones	31
9.1. Informe post-mortem	32
9.1.1. Lo que ha ido bien	32
9.1.2. Lo que ha ido mal	32
9.1.3. Discusión	32
9.2. Trabajos futuros	32
 V Appendices	 33
A. Software Product Lines	35
A.1. Software Product Lines	36
A.2. Feature Models	36
A.3. Automated Analysis of Feature Models	38
A.3.1. Scope	38
A.4. Dynamic Software Product Lines (DSPL)	41
A.5. Hypothesis and Objectives	41
 Referencias bibliográficas	 44

ÍNDICE DE FIGURAS

7.1. Diagrama UML de diseño para la iteración 1	25
A.1. An example of a Home Integration System	37
A.2. A different view on AAFM distinguishing between information extrac- tion and explanatory operations	39

ÍNDICE DE CUADROS

4.1. Tabla resumen de tiempos y planificación	14
4.2. Planificación temporal de iteraciones	14
4.3. Planificación temporal de iteraciones	15
5.1. Tabla resumen de costes	18
7.1. Análisis de valor aportado 0001	24
7.2. Memorando técnico 0001	25
A.1. Most frequently used explanatory operations and their corresponding information extraction operations	43

LISTA DE TAREAS PENDIENTES

Figura: Aquí el modelo de diseño en formato vectorial preferentemente (pdf) . . . 25

■ To Abductive Section in 2.1 36

Figura: A feature model example 37

■ To Abductive Intro 38

PARTE I

INTRODUCCIÓN

CONTEXTO

1

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

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

8

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

1.1 EL MUNDO DEL VIDEOJUEGO

1

El [1, Videojuego] es un ámbito relativamente joven si se compara con otros tipos de entretenimiento o (arte como es considerado por algunos) que han existido a lo largo de la historia contando con apenas 75 años desde la creación de las primeras máquinas que ofrecían una experiencia interactiva que apenas podían considerarse como videojuegos.

2

3

4

5

6

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.

7

8

9

1.2 SUBCONTEXTO

10

1.3 SUBSUBCONTEXTO

11

1.4 ESTADO DEL ARTE

12

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

13

CONTEXTO

1

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

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

R esumen de lo que se explica

2.1 MOTIVACIÓN 1

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 3 4

Se plantea la creación de un videojuego de estrategia por turnos. El jugador podrá escoger un personaje con el que afrontar una Carrera" 5 6

2.2 LISTADO DE OBJETIVOS 7

Objetivo 1. Blabla Detalles del objetivo 1. 8

Objetivo 2. Blabla Detalles del objetivo 2. 9

PARTE II

ORGANIZACIÓN DEL PROYECTO

METODOLOGÍA

1

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

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

8

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

3.1 ESTRUCTURA ORGANIZACIONAL DEL PROYECTO

1

En este caso se presenta un proyecto fin de grado individual de forma que el desarrollo no conlleva ninguna separación de responsabilidades. El estudiante llevará a cabo las labores tanto de desarrollador como de analista y gestor de proyecto. Produciendo así tanto el código a desarrollar como los informes y reportes generados por el desarrollo así como la contabilización y gestión de recursos y horas destinadas al proyecto. ¿Se hace en grupo? En caso afirmativo, ¿cuál va a ser la responsabilidad de cada uno?

2
3
4
5
6
7
8

3.2 METODOLOGÍA DE DESARROLLO

9

La metodología a usar será "Feature-Driven Development". Esta consiste en un proceso de desarrollo de software iterativa e incremental incluida en lo que se conoce en la industria como metodologías ".Agile". Estas metodologías se centran en rasgos generales en aportar valor a los clientes"(en este caso los potenciales compradores de lo que sería el videojuego terminado) primándolo por encima de la generación de documentación, lo cual estaría más enfocado al mantenimiento y mejora de los sistemas que conforman el juego.

10
11
12
13
14
15
16

3.3 DESCRIPCIÓN DETALLADA

17

1. Desarrollo del modelo general: El proyecto comienza con una demostración a alto nivel del contexto y alcance del proyecto. Luego se dividen en sistemas más pequeños para los cuales se generan modelos más detallados. Estos modelos más pequeños se estudian y según su viabilidad se implementan y mezclan con el resto del modelo general.
2. Lista de funcionalidades: El conocimiento recopilado durante esta fase inicial se usa para identificar una lista de características que descomponen funcionalmente cada una de las áreas descritas anteriormente y las divide en piezas que pueden considerarse como unidades de desarrollo.
3. Planear por funcionalidad: Tras completar la lista, el siguiente paso es producir el plan de desarrollo y asignar las funcionalidades a desarrollar como clases a los programadores.

18
19
20
21
22
23
24
25
26
27
28
29

- 1 4. Diseñar por funcionalidad: Un diseño es producido para cada funcionalidad. No
2 debe llevar más de dos semanas desarrollar uno de estos conjuntos de funcionali-
3 dades. Para su desarrollo se generan diagramas de secuencia detallados por cada
4 una de las funcionalidades.
- 5 5. Construir una funcionalidad: Tras una inspección del diseño satisfactoria por ca-
6 da actividad para producir una funcionalidad, el desarrollador reliza su trabajo.
7 Tras las pruebas unitarias y las inspecciones de código, la funcionalidad desarro-
8 llada se mezcla en la rama principal.
- 9 Indicar en qué metodología nos basamos, explicarla brevemente y luego adaptarla
10 a nuestras necesidades. Cada una de estas cuestiones debe ser una subsección.

PLANIFICACIÓN

1

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

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

8

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

4.1 RESUMEN TEMPORAL DEL PROYECTO

1

Resumen del proyecto	
Fecha de inicio	06/07/2024
Fecha de fin	01/10/2024
Periodicidad de las revisiones	3 semanas
Carga de trabajo semanal	24,5 horas
Horas totales previstas	300 horas
Horas finales	234 horas

Cuadro 4.1: Tabla resumen de tiempos y planificación

4.2 PLANIFICACIÓN INICIAL

2

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

3

Resumen de iteraciones	
Iteración 1	06/07/24 a 21/07/24
Iteración 2	22/07/24 a 07/08/24
Iteración 3	08/08/24 a 26/08/24
Iteración 4	27/08/24 a 10/09/24
Iteración 5	11/09/24 a 01/10/24

Cuadro 4.2: Planificación temporal de iteraciones

El criterio seguido para la separación de los periodos de iteración es el siguiente:

4

- La primera guía que se ha tenido en cuenta es la proporcionada en las

5

ESTE CAPÍTULO DEBE ESCRIBIRSE AL COMIENZO DEL PROYECTO

6

4.3 INFORME DE TIEMPOS DEL PROYECTO

7

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

8

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

9

10

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

COSTES

1

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

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

8

R esumen 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

1

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

2

Ya hablaremos de esto

3

5.3 COSTES MATERIALES

4

Y de esto también. Ver Sección §6.2.

5

5.4 COSTES INDIRECTOS

6

Y esto es una fiesta

7

PARTE III

DESARROLLO DEL PROYECTO

ARRANQUE

1

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

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

8

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

6.1 LISTA DE CARACTERÍSTICAS	1
Aplicar aquí la primera iteración de Feature Driven Development.	2
6.2 DISEÑO ARQUITECTÓNICO	3
Descripción de los sistemas de producción, preproducción y pruebas.	4

COSTES

1

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

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

8

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

7.1 CARACTERÍSTICAS A DESARROLLAR

1

1. Funcionalidad A. Ver Tabla §7.1.

2

2. Funcionalidad B.

3

Análisis de valor aportado 0001	
Propuesta	Trabajo que pretende analizarse y justificarse
Valor	Qué valor aporta al proyecto o al usuario final.
Coste	Qué costes en términos de esfuerzo, adquisiciones y limitaciones tiene la propuesta
Opciones	Qué otras opciones se tienen que aporten un valor similar? ¿Es realmente un valor relevante para el proyecto/cliente
Riesgos	Qué riesgos pueden surgir a la hora de desarrollar esta propuesta.
Deuda técnica	Posibles deudas técnicas que se asumen con el desarrollo de esta propuesta.

Cuadro 7.1: Análisis de valor aportado 0001

7.2 DISEÑO

4

Aquí una discusión de cómo va a afectar todo al diseño

5

Debe insertarse un diagrama UML de diseño con los cambios y hacer referencia en el texto así Fig. §7.1.

6

7

Un memorando técnico por cada decisión de diseño.

8

7.3 IMPLEMENTACIÓN

9

Un memorando técnico por cada decisión de implementación y refactorización que afecte al diseño.

10

11

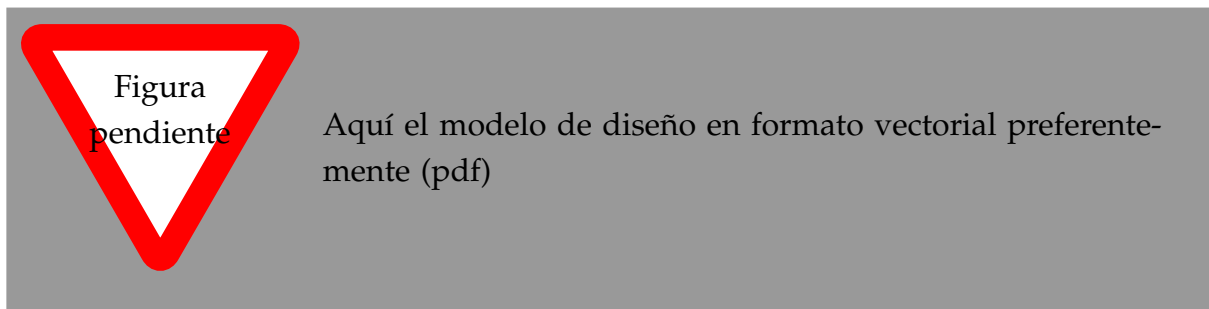


Figura 7.1: Diagrama UML de diseño para la iteración 1

Memorando técnico 0001	
Asunto	¿Cuál es el problema?
Resumen	¿Cuál es la solución propuesta?
Factores causantes	Descripción pormenorizada del problema
Solución	Descripción pormenorizada de la solución propuesta
Motivación	¿Por qué propone esta solución?
Cuestiones abiertas	Factores a tener en cuenta en la solución cuya dimensión se reconoce.
Alternativas	Otras soluciones consideradas y la razón por la que se excluyeron.

Cuadro 7.2: Memorando técnico 0001

Identificador		Descripción de la acción de alto nivel		
0001		Prueba		
Métodos de alto nivel				
[return_type] method_name1 (param1:type1, ...)				
Pasos (Usar Pseudocódigo o similar)				
1. Paso 1.				
2. Paso 2.				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	ClassName	[return_type] method_name1 (param1:type1, ...)	001	SI
Diagrama de Colaboración				

Identificador	Descripción de la acción de alto nivel			
alvotermar02	Grubber			
Métodos de alto nivel				
[return_type] grubber (param1:type1, ...)				
Pasos (Usar Pseudocódigo o similar)				
1. Lanzar 2 dados				
2. Compara resultado de los dados con kicking del open-side				
2.1. Si valor dados es menor o igual a kicking, avanza 10m				
3.1. Si no hay defensa y el golpeo es exitoso, el pateador retiene la posesión del balón				
3.2. Si hay defensa y el golpe es exitoso, el atacante tira un dado y suma su valor al de speed y strength y el defensor lanza 2 dados y lo suma al valor de speed y strength de su jugador, el vencedor será aquel que tenga más puntos, si es igual, la posesión es del defensor				
4.1. Si no es exitoso y hay defensa el balón pasa a posesión del defensor				
4.2. Si no es exitoso y no hay defensa de lanza un line-out				
Métodos de bajo nivel necesarios				
Paso	Clase	Método	Mem. Técn.	IU
1	Dice	[Integer] throwDice ()	001	SI
2	ClassName	[Int] compareKickingToDice (kicking:Integer, dice: Integer)	001	SI
2.1	ClassName	[Integer] setLine (line:Integer)	001	SI
4.2	ClassName	[Integer] lineOut ()	001	SI

2

7.4 PRUEBAS

3

Descripción de las pruebas realizadas al software

4

7.5 DESPLIEGUE

5

Breve resumen de cómo se han desplegado los cambios en el sistema de producción.

6

7

PARTE IV

CIERRE DEL PROYECTO

MANUAL DE USUARIO

1

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

4

Ernest Hemingway (1899–1961),

5

Novelist

6

7

8

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

8.1 SECCIÓN LIBRE	1
Estructurar en función del proyecto.	2

CONCLUSIONES

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*

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

9.1 INFORME POST-MORTEM	1
Qué es un informe post-mortem	2
9.1.1 Lo que ha ido bien	3
▪ Argumento a favor 1.	4
▪ Argumento a favor 2.	5
▪ Argumento a favor 3.	6
9.1.2 Lo que ha ido mal	7
▪ Argumento en contra 1.	8
▪ Argumento en contra 2.	9
▪ Argumento en contra 3.	10
9.1.3 Discusión	11
En función de lo anterior, qué cambiaría si empezara hoy el proyecto de nuevo.	12
9.2 TRABAJOS FUTUROS	13
Enumera los puntos abiertos y que no se han resuelto. Indica si darían lugar a otro proyecto y de qué forma se podría acotar.	14
	15

PARTE V

APPENDICES

SOFTWARE PRODUCT LINES

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) [?]
- 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.

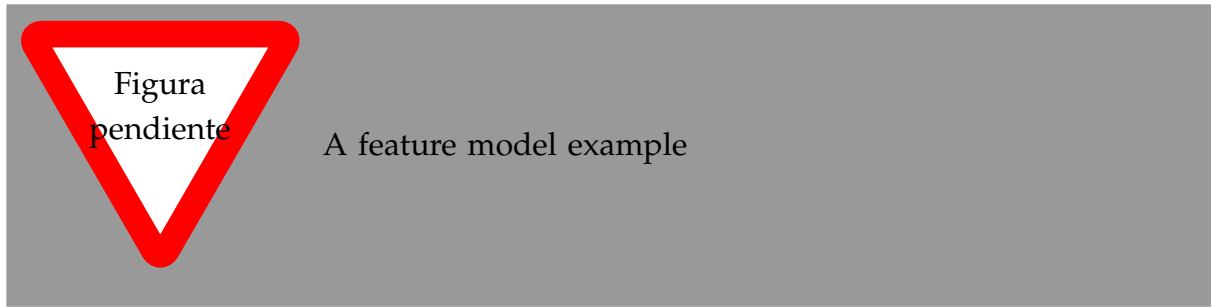


Figura A.1: An example of a Home Integration System

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, $S_P = \{\dots\}$, $R_P = \{\dots\}$ and $U_P = \{\dots\}$ define a partial configuration for the sample FM where some features are still to be decided if they are to be selector 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, $S_F = \{\dots\}$ and $R_F = \{\dots\}$ 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) [?]. 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

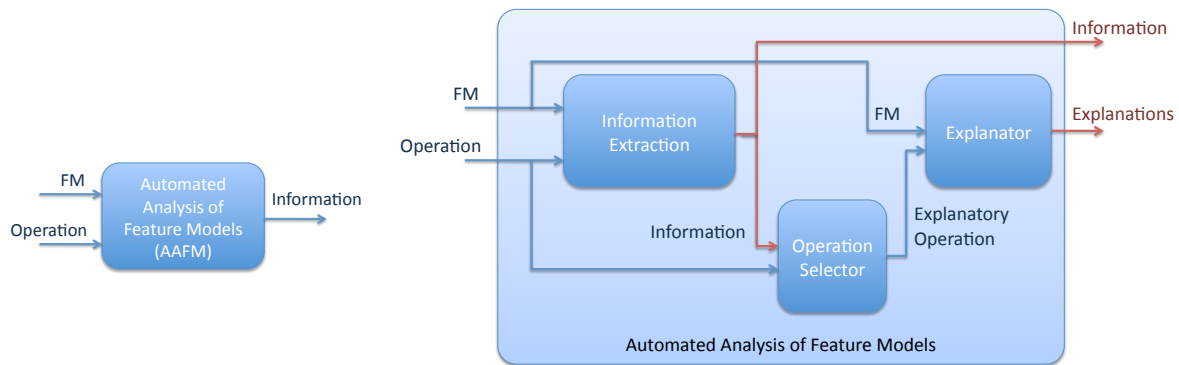
A.3.1 Scope

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. [?]

A Catalog of FM Explanatory Operations (2009 version)		
Information Extraction Operation	FM Explanatory Operations	
	<i>Why? operation</i>	<i>Why not? operation</i>
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	
	<i>Why? operation</i>	<i>Why not? operation</i>
Valid Configuration	valid partial conf.	invalid partial conf.

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

BIBLIOGRAFÍA

- [1] Wikipedia. Videojuego — Wikipedia, the free encyclopedia. <http://es.wikipedia.org/w/index.php?title=Videojuego&oldid=161053303>, 2024. [Online; accessed 07-July-2024]. (page 4).