



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

A cotutelle thesis submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science at the Universitat Politècnica de
València and the University of Sousse

Towards designing and generating user interfaces by using expert knowledge

*Defended on the 8th of November 2022 by
Amani Braham*

Supervisors

Prof. Félix Buendia Garcia Polytechnic University of Valencia , Spain
Prof. Faiez Gargouri University of Sfax, Tunisia
Prof. Maha Khemaja University of Sousse, Tunisia

Members of the thesis committee

Prof. Sonia Ghannouchi University of Sousse, Tunisia President
Prof. Lilia Cheniti University of Sousse, Tunisia Secretary
Prof. Leila Ben Ayed University of Mannouba, Tunisia Speaker

External reviewers of the thesis

Prof. Jean Vanderdonckt Université Catholique de Louvain, Belgium
Prof. José Ignacio Panach Navarrete University of Valencia, Spain
Prof. Martin Hitz Alpen-Adria-Universität Klagenfurt, Austria

Dedication

*To my family,
who have provided me with
support and encouragement
throughout my studies.*

Acknowledgement

This thesis has been a long but fulfilling journey. This journey would not have been possible to complete without the invaluable support and continuous encouragement of several who have contributed to the success of my PhD.

I would like to express my gratitude to my supervisors, Mr. Félix Buendia Garcia, Mr. Faeiz Gargouri and Ms. Maha Khemaja, for their valuable insights and the enormous patience that was necessary for guiding me all along the way. Their support, advice and feedback have been a fundamental part to get to the end of this journey.

I want to extend my gratitude to all the members of the Computing Engineering Department (DISCA) of the Universitat Politecnica de Valencia (UPV) for the excellent working environment they provide throughout my journey. In addition, I am grateful for the academic support of the University of Sousse, the Higher Institute of Computer Science and Communication Technologies (ISITCom) and the PRINCE Laboratory.

I also want to pay tribute to the members of the jury for agreeing to be a part of the court and the external reviewers for their thoughtful comments and positive feedback.

A very special thanks goes out to Roua. We started our university journey at the same time and she has been my friend and support from that moment until now.

Finally, thanks to Yosra and Khadija, my two sisters, and my brother Ali for their love and for keeping me always smiling. I have to appreciate the irreplaceable support of my parents, Ahmed and Saida, who have given me unstinting love, care, trust, help, and encouragement.

Amani Braham

English Summary

The research reported in the present PhD dissertation is conducted through the design science methodology that focuses on creating and evaluating artifacts. In the current thesis, the main artifact is the novel approach to design and generate user interfaces using expert knowledge. In order to enable the use of expert knowledge, the present approach is devoted to reuse design patterns that incorporate expert knowledge of interface design and provide reusable solutions to various design problems. The main goal of the proposed approach is to address the use of design patterns in order to ensure that expert knowledge is integrated into the design and generation of user interfaces for mobile and Web applications. The specific contributions of this thesis are summarized below:

This first contribution is the AUIDP framework that is defined to support the design and generation of adaptive interfaces for Web and mobile applications using HCI design patterns. The proposed framework spans over design-time and run-time. At design-time, models of design patterns along with user interface and user profile are defined following a specific development methodology. At run-time, the created models are used to allow the selection of HCI design patterns and to enable the generation of user interfaces from the design solutions provided by the relevant design patterns.

The second contribution is a specification method to establish an ontology model that turns traditional text-based representation into formal HCI design pattern representation. This method adopts the Neon methodology to achieve the transition from informal to formal representations. The created ontology model is named MIDEP, which is a modular ontology that captures knowledge about design patterns as well as the user interface and user's profile.

The third contribution is the IDEPAR, which is the first system within the global AUIDP framework. This system aims to automatically recommend the most relevant design patterns for a given design problem. It is based on a hybrid approach that relies on a mixed combination of text-based and ontology-based recommendation techniques to produce design pattern recommendations that provide appropriate design solutions.

The fourth contribution is an interface generator system called ICGDEP, which is proposed to automatically generate the user interface source code for Web and mobile applications. The proposed ICGDEP is the second system within the global AUIDP framework and relies on the use of HCI design patterns that are recommended by the IDEPAR system. It mainly aims at automatically generating the user interface source code from the design solutions provided by design patterns. To achieve this, the ICGDEP system is based on a generation method that allows the generation of user interface source code for the target application.

The contributions provided in the present thesis have been validated through different perspectives. First, the evaluation of the developed MIDEP ontology is performed using competency questions, technology-based, and application-based evaluation approaches. Second, the evaluation of the IDEPAR system is established through an expert-based gold standard and a user-centric evaluation study. Then, the ICGDEP system is evaluated in terms of being effectively used by developers, considering the productivity factor. Finally, the evaluation of the global AUIDP framework is conducted through case studies and usability studies.

The obtained results demonstrate: (i) The capability of the proposed specification method to produce a correct and effective ontology model. (ii) The efficiency of the IDEPAR system to recommend the most relevant HCI design patterns and the positive user experience regarding the recommended patterns. (iii) The feasibility of the ICGDEP system to automate the generation of user interface source code and to hasten the development process by reducing the development time. (iv) The ability of the AUIDP framework to carry out run-time user interface adaptations and to generate usable interfaces that are accepted by end users.

Resumen Espanol / Spanish Summary

La investigación reportada en la presente tesis doctoral se lleva a cabo a través de la metodología de la ciencia del diseño que se centra en la creación y evaluación de artefactos. En esta tesis, el principal artefacto es el novedoso enfoque para diseñar y generar interfaces de usuario utilizando el conocimiento experto. Con el fin de permitir el uso del conocimiento experto, el enfoque propuesto se basa en la reutilización de patrones de diseño que incorporan el conocimiento experto del diseño de la interfaz y proporcionan soluciones reutilizables a diversos problemas de diseño. El objetivo principal de dicho enfoque es abordar el uso de patrones de diseño a fin de garantizar que los conocimientos especializados se integren en el diseño y la generación de interfaces de usuario para aplicaciones móviles y web. Las contribuciones específicas de esta tesis se resumen a continuación:

Una primera contribución consiste en el marco AUIDP que se define para apoyar el diseño y la generación de interfaces adaptativas para aplicaciones web y móviles utilizando patrones de diseño HCI. El marco propuesto abarca tanto la etapa de diseño como la de ejecución de dichas interfaces. En el momento del diseño, los modelos de patrones de diseño junto con la interfaz de usuario y el perfil de usuario se definen siguiendo una metodología de desarrollo específica. En tiempo de ejecución, los modelos creados se utilizan para permitir la selección de patrones de diseño de HCI y para permitir la generación de interfaces de usuario a partir de las soluciones de diseño proporcionadas por los patrones de diseño relevantes.

La segunda contribución es un método de especificación para establecer un modelo de ontología que convierte la representación tradicional basada en texto en la representación formal del patrón de diseño de HCI. Este método adopta la metodología Neon para lograr la transición de las representaciones informales a las formales. El modelo de ontología creado se llama MIDEP, que es una ontología modular que captura el conocimiento sobre los patrones de diseño, así como la interfaz de usuario y el perfil del usuario.

La tercera contribución es el IDEPAR, que es el primer sistema dentro del marco global del AUIDP. Este sistema tiene como objetivo recomendar automáticamente los patrones de diseño más relevantes para un problema de diseño dado. Se basa en un enfoque híbrido que utiliza una combinación mixta de técnicas de recomendación basadas en texto y ontología para producir recomendaciones de patrones de diseño que proporcionan soluciones de diseño apropiadas.

La cuarta contribución es un sistema generador de interfaz llamado ICGDEP, que se propone para generar automáticamente el código fuente de la interfaz de usuario para

aplicaciones web y móviles. El ICGDEP es el segundo sistema dentro del marco global de AUIDP y se basa en el uso de patrones de diseño de HCI que son recomendados por el sistema IDEPAR. Su objetivo principal es generar automáticamente el código fuente de la interfaz de usuario a partir de las soluciones de diseño proporcionadas por los patrones de diseño. Para lograr esto, el sistema ICGDEP utiliza un método que permite la generación de código fuente de interfaz de usuario para la aplicación de destino.

Las contribuciones aportadas en la presente tesis han sido validadas a través de diferentes perspectivas. En primer lugar, la evaluación de la ontología MIDEP desarrollada se realiza utilizando preguntas de competencia, enfoques de evaluación basados en la tecnología y basados en aplicaciones. En segundo lugar, la evaluación del sistema IDEPAR se establece mediante un patrón producido por expertos y un estudio de evaluación centrado en el usuario. Luego, el sistema ICGDEP es evaluado en términos de ser utilizado efectivamente por los desarrolladores, considerando el factor de productividad. Por último, la evaluación del marco mundial de AUIDP se lleva a cabo mediante estudios de casos y estudios de usabilidad.

Los resultados obtenidos demuestran: (i) La capacidad del método de especificación propuesto para producir un modelo ontológico correcto y efectivo. (ii) La eficiencia del sistema IDEPAR para recomendar los patrones de diseño de HCI más relevantes y la experiencia positiva del usuario con respecto a los patrones recomendados. (iii) La viabilidad del sistema ICGDEP para automatizar la generación de código fuente de interfaz de usuario y acelerar el proceso de desarrollo mediante la reducción del tiempo de desarrollo. (iv) La capacidad del marco de AUIDP para llevar a cabo adaptaciones de interfaz de usuario en tiempo de ejecución y para generar interfaces que sean aceptadas por los usuarios finales.

Resum Valenciano / Valencian Summary

La investigació reportada en aquesta tesi doctoral es duu a terme a través de la metodologia de la ciència del disseny que se centra en la creació i avaluació d'artefactes. En aquesta tesi, el principal artefacte és el nou enfocament per dissenyar i generar interfícies d'usuari utilitzant el coneixement expert. Per tal de permetre l'ús del coneixement expert, l'enfocament proposat es basa en la reutilització de patrons de disseny que incorporen el coneixement expert del disseny de la interfície i proporcionen solucions reutilitzables a diversos problemes de disseny. L'objectiu principal d'aquest enfocament és abordar l'ús de patrons de disseny per tal de garantir que els coneixements especialitzats s'integrin en el disseny i la generació d'interfícies d'usuari per a aplicacions mòbils i web. Les contribucions específiques d'aquesta tesi es resumeixen a continuació:

Una primera contribució consisteix en el marc AUIDP que es defineix per donar suport al disseny i generació d'interfícies adaptatives per a aplicacions web i mòbils utilitzant patrons de disseny HCI. El marc proposat inclou tant l'etapa de disseny com la d'execució de les interfícies esmentades. En el moment del disseny, els models de patrons de disseny juntament amb la interfície d'usuari i el perfil d'usuari es defineixen seguint una metodologia de desenvolupament específica. En temps d'execució, els models creats s'utilitzen per permetre la selecció de patrons de disseny de HCI i per permetre la generació de interfícies d'usuari a partir de les solucions de disseny proporcionades pels patrons de disseny rellevants.

La segona contribució és un mètode d'especificació per establir un model d'ontologia que converteix la representació tradicional basada en text en la representació formal del patró de disseny de HCI. Aquest mètode adopta la metodologia Neon per aconseguir la transició de les representacions informals a les formals. El model d'ontologia creat s'anomena MIDEP, una ontologia modular que captura el coneixement sobre els patrons de disseny, així com la interfície d'usuari i el perfil de l'usuari.

La tercera contribució és l'IDEPAR, que és el primer sistema dins del marc global de l'AUIDP. Aquest sistema té com a objectiu recomanar automàticament els patrons de disseny més rellevants per a un problema de disseny donat. Es basa en un enfocament híbrid que utilitza una combinació mixta de tècniques de recomanació basades en text i ontologia per produir recomanacions de patrons de disseny que proporcionen solucions de disseny apropiades.

La quarta contribució és un sistema generador d'interfície anomenat ICGDEP, que es proposa per generar automàticament el codi font de la interfície d'usuari per a aplicacions web i mòbils. L'ICGDEP és el segon sistema dins del marc global d'AUIDP i es basa en l'ús de

patrons de disseny de HCI que són recomanats pel sistema IDEPAR. El seu objectiu principal és generar automàticament el codi font de la interfície d'usuari a partir de les solucions de disseny proporcionades pels patrons de disseny. Per aconseguir-ho, el sistema ICGDEP utilitza un mètode que permet generar codi font d'interfície d'usuari per a l'aplicació de destinació.

Les contribucions aportades a la present tesi han estat validades a través de diferents perspectives. En primer lloc, l'avaluació de l'ontologia MIDEP desenvolupada es fa utilitzant preguntes de competència, enfocaments d'avaluació basats en la tecnologia i basats en aplicacions. En segon lloc, l'avaluació del sistema IDEPAR s'estableix mitjançant un patró produït per experts i un estudi d'avaluació centrat en l'usuari. Després, el sistema ICGDEP és avaluat en termes de ser utilitzat efectivament pels desenvolupadors, considerant el factor de productivitat. Finalment, l'avaluació del marc mundial d'AUIDP es fa mitjançant estudis de casos i estudis d'usabilitat.

Els resultats obtinguts demostren: (i) La capacitat del mètode d'especificació proposat per produir un model ontològic correcte i efectiu. (ii) L'eficiència del sistema IDEPAR per recomanar els patrons de disseny de HCI més rellevants i l'experiència positiva de l'usuari pel que fa als patrons recomanats. (iii) La viabilitat del sistema ICGDEP per automatitzar la generació de codi font d'interfície d'usuari i accelerar el procés de desenvolupament mitjançant la reducció del temps de desenvolupament. (iv) La capacitat del marc d'AUIDP per dur a terme adaptacions d'interfície d'usuari en temps d'execució i per generar interfícies acceptades pels usuaris finals.

Contents

Contents	xi
Acronyms	xv
List of Figures	xviii
List of Tables	xxi
List of Algorithms	1
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Objectives and Research Questions	4
1.4 Thesis Contribution	5
1.5 Research Methodology	6
1.5.1 Methodological framework	6
1.5.2 Methodology applied to this thesis	6
1.6 Thesis Outline	9
2 Background	11
2.1 Introduction	11
2.2 Fundamentals of Design Patterns	11
2.2.1 Design pattern concept	11
2.2.2 Design pattern domain	12
2.2.3 Design pattern documentation	13
2.2.4 Design pattern language	13
2.2.5 Design pattern specification	14
2.3 Overview of Ontologies	14
2.3.1 Ontology definition	15
2.3.2 Ontology components	15
2.3.3 Methodologies for designing ontologies	16
2.3.4 Ontology development tools	19
2.3.5 Ontology evaluation	19
2.4 Concluding Remarks	23
3 State of the Art	25
3.1 Introduction	25

3.2	User Interface Specification Approaches	26
3.2.1	Review of user interface specification	26
3.2.2	Analysis and discussion	29
3.3	Design Pattern Recommender Systems	30
3.3.1	Review of design pattern recommendation	30
3.3.2	Comparative analysis	31
3.4	User Interface Adaptation and Generation Approaches	34
3.4.1	Review of user interface adaptation and generation	34
3.4.2	Comparative analysis	35
3.5	Concluding Remarks	37
4	AUIDP: A Framework for the Design and Generation of User Interfaces	41
4.1	Introduction	41
4.2	Main Building Blocks	41
4.3	Framework Architecture	43
4.4	Framework Implementation	45
4.4.1	Implementation overview	46
4.4.2	Technical architecture of the AUIDP framework	46
4.5	Concluding Remarks	48
5	Design Pattern Specification Method	51
5.1	Introduction	51
5.2	Design Pattern Specification Overview	51
5.3	Ontology Development Phases	55
5.3.1	Specification	55
5.3.2	Scheduling	55
5.3.3	Knowledge resource reuse and re-engineering	55
5.3.4	Ontology design	60
5.3.5	Ontology implementation	64
5.4	Concluding Remarks	65
6	Design Pattern Recommender System	69
6.1	Introduction	69
6.2	Recommender System Overview	69
6.3	Recommender System Architecture	69
6.3.1	NLP module	71
6.3.2	Semantic module	72
6.4	IDEPAR Implementation	75
6.4.1	Server-side implementation	75
6.4.2	Client-side implementation	76
6.4.3	Design pattern recommendation example	76
6.4.4	IDEPAR prototype	79
6.5	Concluding Remarks	82
7	User Interface Generator System	85
7.1	Introduction	85
7.2	System Architecture	85
7.2.1	Pattern instantiation	86

7.2.2	Pattern integration	87
7.2.3	User interface generation	92
7.3	ICGDEP Implementation	98
7.3.1	ICGDEP tool	98
7.3.2	User interface generation example	99
7.4	Concluding Remarks	100
8	Evaluation	105
8.1	Introduction	105
8.2	Evaluation of the MIDEP Ontology	105
8.2.1	Evaluation overview	105
8.2.2	Competency questions evaluation	106
8.2.3	Technology-based evaluation	109
8.2.4	Application-based evaluation	111
8.2.5	Discussion	114
8.3	Evaluation of the IDEPAR System	114
8.3.1	Study 1: Expert-based gold standard evaluation	115
8.3.2	Study 2: User-centric evaluation	118
8.3.3	Discussion	123
8.4	Evaluation of the ICGDEP System	125
8.4.1	Experimental settings	125
8.4.2	Results and discussion	126
8.5	Evaluation of the Global AUIDP Framework	127
8.5.1	Case Studies	127
8.5.2	Usability study	129
8.5.3	Discussion	132
8.6	Concluding Remarks	132
9	Conclusions and Future Works	135
9.1	Introduction	135
9.2	Summary of Contributions	135
9.3	Scientific Results	136
9.4	Future Works	138
9.5	Concluding Remarks	139
	Appendices	141
	HCI Design Pattern Catalog	143
	Post-test Questionnaire	147
	Usability Testing Questionnaire	149
	Bibliography	151

Acronyms

UI	User Interface
IT	Information Technology
HCI	Human Computer Interaction
MDD	Model-Driven Development
UsiXML	USer Interface eXtensible Markup Language
MARIA	Model-based lLanguage foR Interactive Applications
IFML	Interaction Flow Modeling Language
IS	Information Systems
XML	eXtensible Markup Language
WSML	Web Service Modeling Language
RDF	Resource Description Framework
RDFS	RDF Schema
OWL	Ontology Web language
MDA	Model-Driven Architecture
MBUID	Model-Based User Interface Development
CRF	CAMELEON Reference Framework
FUI	Final User Interface
AUI	Abstract User Interface
CUI	Concrete User Interface
TERESA	Transformation Environment for inteRactivE Systems representations
GUI	Graphical User Interface
CTT	Concur Task Tree
PIM	Platform Independent Model

JUST-UI	JUST-User Interface
PIM tool	Patterns In Modeling tool
PaMGIS	Pattern-Based Modeling and Generation of Interactive System
PD-MBUI	Pattern-Driven and Model-Based User Interface
PLML	Pattern Language Markup Language
XPLML	eXtended Pattern Language Markup Language
UIPLML	User Interface Pattern Language Markup Language
XUL	XML User interface Language
GoF	Gang-of-Four
VSM	Vector Space Model
CS	Cosine Similarity
SOA	Service-Oriented Architecture
NLP	Natural Language Processing
A-UI/UX-A	Adaptive User Interface User Experience Authoring
EMF	Eclipse Modeling Framework
DSL	Domain Specific Languages
AUIDP	Adaptive User Interface Design Pattern
IDEPAR	user Interface Design Pattern Recommender
ICGDEP	user Interface Code Generator using DDesign Patterns
MIDEP	Modular user Interface DDesign Pattern
ORSD	Ontology Specification Requirement Document
NORs	Non-Ontological Resources
POS	Part Of Speech
SSM	Semantic Similarity Measures
onto-UIAR	onto User Interface Adaption Rules
PSM	Platform Specific Model
TP	True Positive
FP	False Positive
TN	True Negative

- FN** False Negative
- GS** Gold Standard
- SD** Standard Deviation
- SUS** System Usability Scale
- QUIS** Questionnaire for User Interaction Satisfaction
- TCT** Task Completion Time

List of Figures

1.1	Research methodology overview of this thesis	7
2.1	Design pattern structure	12
2.2	Overview of ontology components	16
2.3	Ontology evaluation based on the definition of ontologies	20
3.1	Research aspects	25
4.1	Main building blocks of the AUIDP Framework	42
4.2	Overview of the modeling stage	43
4.3	Architecture of the AUIDP framework	44
4.4	Components of the IDEPAR system	45
4.5	Components of the ICGDEP system	45
4.6	Overview of the framework implementation	46
4.7	Execution environment for mobile applications	49
4.8	Execution environment for Web applications	50
5.1	Scenarios for building the MIDEP ontology	53
5.2	MIDEP ontology life cycle	58
5.3	MIDEP ontology modules	61
5.4	Conceptual model of the Design Pattern module	62
5.5	Conceptual model of the User Interface module	63
5.6	Conceptual model of the User Profile module	64
5.7	Main characteristics of the MIDEP ontology	65
5.8	Screenshot of OWL ontology classes implemented using Protégé	66
5.9	Screenshot of OWL ontology object properties, data properties, and instances implemented using Protégé	67
6.1	Overview of the IDEPAR system	70
6.2	Architecture of the IDEPAR system	70
6.3	NLP module overview	71
6.4	Semantic module overview	72
6.5	Design problem ontology model	73
6.6	Semantic Similarity Check Algorithm	74
6.7	IDEPAR system: server-side and client-side implementation	76
6.8	Splitter results for DPS-1	77
6.9	Tokenizer results for DPS-1	77
6.10	POS tagger results for DPS-1	77
6.11	Classifier results for DPS-1	78

6.12	NLP module results for DPS-1	78
6.13	Semantic module results for DPS-1	79
6.14	IDEPAR application main interface	80
6.15	Selection of design problem DPS-1	81
6.16	List of recommended design patterns for DPS-1	81
6.17	Selection of design problem DPS-2	82
6.18	List of recommended design patterns for DPS-2	82
7.1	ICGDEP system architecture	86
7.2	Pattern instantiation overview	86
7.3	SPARQL query for extracting pattern fragments	87
7.4	Pattern integration overview	87
7.5	Workflow for refining pattern fragments	88
7.6	Design solution preprocessing major tasks	89
7.7	GATE application pipeline for semantic annotation	90
7.8	Jape rule example for ontology instance annotation	90
7.9	IF-THEN rule example	91
7.10	Inference rule example	91
7.11	Assemble pattern fragment into UI model overview	92
7.12	User interface generation main phases	92
7.13	UI model adaptation overview	93
7.14	Onto-UIAR ontological model	93
7.15	Model transformation overview	94
7.16	OntoPIM Transformation rules	95
7.17	Source code generation overview	96
7.18	GUI meta-model	97
7.19	MIDEPDSL grammar: Application rule	97
7.20	MIDEPDSL grammar: Component Element rule	97
7.21	Extract of the transformation template	98
7.22	ICGDEPTool architecture	99
7.23	UI generation example for DPS-1: Results of the pattern instantiation component	100
7.24	UI generation example for DPS-1: Results of the pattern integration component	101
7.25	UI generation example for DPS-1: Results of the UI generation component	102
7.26	UI generation example for DPS-2	103
8.1	SPARQL query results for CQ1	107
8.2	SPARQL query results for CQ2	107
8.3	SPARQL query results for CQ3	108
8.4	SPARQL query results for CQ4	108
8.5	SPARQL query results for CQ5	109
8.6	OOPS! evaluation summary before corrections	110
8.7	OOPS! evaluation summary after corrections	112
8.8	Interaction design pattern category	113
8.9	Example of design pattern description	113
8.10	RESTful Web service result: (a) before ontology population, (b) after ontology population	114
8.11	Representative questions from each ResQue layer	119
8.12	Distribution of answers to post-test questionnaire: Perceived system quality layer	121

8.13	Distribution of answers to post-test questionnaire: Belief layer	122
8.14	Distribution of answers to post-test questionnaire: Attitude layer	122
8.15	Distribution of answers to post-test questionnaire: Behavior layer	122
8.16	Results of the Cronbach's alpha coefficient	123
8.17	Scenario example	125
8.18	Screenshots of the generated user interfaces	126
8.19	Case study 1	128
8.20	Case study 2	129
8.21	Case study 3	130
8.22	Interfaces for event scheduling: (a) Traditional interface, (b) Adaptive interface	131

List of Tables

2.1	Design pattern documentation forms	13
3.1	Comparison of existing design pattern recommendation systems	33
3.2	Summary of the surveyed UI adaptation and generation approaches	38
5.1	Excerpt of the MIDEP Ontology Requirement Specification Document: Part I .	56
5.2	Excerpt of the MIDEP Ontology Requirement Specification Document: Part II	57
5.3	Results of the NOR from Websites	59
5.4	Results of the NOR from the scientific literature	59
5.5	Concepts dictionary of the Design Pattern module	62
5.6	Concepts dictionary of the User Interface module	63
5.7	Concepts dictionary of the "User Profile" module	64
6.1	A rule example for matching design pattern groups.	75
7.1	Onto-UIAR concept description	94
7.2	Some transformation rules for Web (Angular) and mobile (Ionic) applications .	98
8.1	MIDEP ontology pitfalls detected by OOPS!	111
8.2	Overview of the design problem scenarios	117
8.3	List of HCI design patterns selected by expert	117
8.4	Performance results	118
8.5	Pre-study questionnaire results	120
8.6	Post-test questionnaire results	121
8.7	Correlations between participants' knowledge about recommender systems and answers of Q7 and Q10	124
8.8	Correlations between participants' level of expertise with HCI design patterns and answers of Q1, Q7, Q9, and Q10	124
8.9	Statistical results of satisfaction and learnability dimensions	132
9.1	List of the contributions and publications achieved	138
1	Description of HCI design patterns for "FontColor" group	143
2	Description of HCI design patterns for "Background" group	143
3	Description of HCI design patterns for "FontSize" group	144
4	Description of HCI design patterns for "InputMode" group	144
5	Description of HCI design patterns for "BasicInteraction" group	144
6	Description of HCI design patterns for "Zoom" group	144
7	Description of HCI design patterns for "Navigating" group	145

8	Description of HCI design patterns for "OutputMode" group	145
9	Description of HCI design patterns for "MakingChoice" group	145
10	Description of HCI design patterns for "Shopping" group	146
11	Questions of the Post-test Questionnaire	147
12	Questions of the Usability Testing Questionnaire	149

Introduction

In the present PhD thesis, we propose an approach that considers the use of expert knowledge in order to address the design and generation of adaptive user interfaces for Web and mobile applications. In addition, this work deals with the development of a framework that supports the proposed approach. Throughout this chapter, we describe in details the motivation, objectives and contributions of the present thesis.

This chapter incorporates six sections. Section 1.1 details the motivation behind this thesis and introduces the scientific and technical challenges of User Interface (UI) design and generation. Section 1.2 presents the problems that this thesis attempts to solve. Section 1.3 details objectives to be achieved and states the research questions addressed in this thesis. Section 1.4 describes the main contributions covered by this research. Section 1.5 provides the research methodology used to develop this thesis. Finally, this chapter ends describing the general overview of the rest of the present PhD dissertation.

1.1 Motivation

Information Technology (IT) has gained much attention around the world. The continuous advance in the development of IT systems has recently witnessed a rapid growth of platforms, devices and environments [Ruiz et al., 2019]. This trendy movement is observed by the high proliferation of interaction devices that has brought about a shift in the way people live and interact with each other. This shift is allowing users to be surrounded by a broad range of mobile devices (e.g. smartphones, tablets) capable of providing interfaces for conducting their everyday activities [Rodrigues et al., 2011]. Such devices are used by 86% of people across the globe [Gadasin et al., 2020] and it is predicted that more than 90% of adults in developed countries will own at least one mobile device by the end of 2023 [Wang et al., 2018].

The intensive penetration of mobile devices has fueled a new wave of demand for mobile and Web applications. The extensive use of these devices makes the application industry a multi-billion dollar industry [ABIResearch, 2013]. Such rapidly growing demands call for the development of new applications that continuously improve user experiences by exploiting contextual information from the devices surrounding users. The mobility

exhibited by existing devices can enable users to use applications everywhere. As a result, the surrounding context of these interaction devices keeps changing over time. By such context awareness, applications need to be adaptive in the sense that they are able to dynamically adapt to context changes at run-time. These applications are intended to be used by various users with different profiles, needs, and devices that raise new challenges as users want to have UIs that meet their requirements [Soui et al., 2017].

Users usually perceive the quality of the UI as the quality of the overall application. This has promoted an increase in design possibilities and a growing interest in the study of UIs [Gomaa et al., 2005] within the Human Computer Interaction (HCI) research community to satisfy users' expectations.

The UI is a crucial component for the acceptance of the whole application. The acceptance and the usability of a UI are often strongly influenced by the context changes, making software developers and UI designers facing new challenges to design and generate adaptive UIs that support the variability of the current context in terms of the platform, the user, and the environment [Calvary et al., 2003]. Adaptive UIs are supposed to adapt interaction contents and information processing modes automatically to meet the changing context, users' needs and disabilities at any time [Letsu-Dake and Ntuen, 2009]. They have been introduced as a solution to address context variability due to their ability to automatically adapt at run-time to meet the current context [Akiki et al., 2014] and to enable their adaptation according to the UIs types and the user's choice [Jellad and Khemaja, 2014]. Recently, adaptive UIs have made tremendous progress regarding the big evolution of technology. This fact makes their development task even more complex and it requires extra knowledge and expertise. The development of these UIs is a time consuming and error prone task that reaches 48% of the total application code and 50% of the development time [Myers and Rosson, 1992, Kennard and Leaney, 2010]. Thus, the application development cost is strongly influenced by the effort devoted to the UIs [Macik, 2012]. The complexity and the effort time needed for the development of UIs is due to the rapid growth in the manufacturing of technological devices [Petrasch, 2007]. Further complexity arises to meet the dynamicity of the context of use. The development of various UIs for the same functionality in different contexts of use is not a trivial task since context changes. This leads to diverse adaptations involving high cost incurred by manually developing different versions of UIs [Akiki et al., 2014]. As a result, the design and the development of adaptive UIs requires appropriate methods.

1.2 Problem Statement

The development process of UIs, ranging from early design to coding and generation, is a complex set of activities that require various stakeholders from different disciplines. The user is the principal stakeholder who interacts directly with the UI, so well-designed UIs can determine the application's appeal to a user.

Developing separate UIs for each user's profile is neither feasible nor a cost effective solution, especially when we consider the variability of the context of use. Providing UIs that meet the dynamics of the current context is a complex task, which addresses special challenges beyond traditional UI development. Moreover, the increase in the complexity of adaptive UIs along with the diversity of interaction modalities impose new requirements for automated methods. One promising solution to deal with the complexity of adaptive

UI development is Model-Driven Development (MDD) approaches. These approaches provide a way to decrease the development effort by using high-level UI models. In this sense, various existing approaches support the transformation of high level UI models to the source code of the final UI. Most popular approaches are based on User Interface eXtensible Markup Language (UsiXML) [Limbourg et al., 2004], Model-based Language for Interactive Applications (MARIA) [Paterno' et al., 2009], and Interaction Flow Modeling Language (IFML) [Brambilla and Fraternali, 2014]. Besides, there exist other approaches including Supple [Gajos et al., 2010] and MyUI [Peissner et al., 2012] that provide tools and techniques for developing adaptive UIs using predefined rules at run-time. Despite the fact that these approaches provide various implementations of UIs, the generation of adaptive UIs is not usually addressed at run-time since existing approaches generate the UI at design-time and adapt the UI only for predefined situations at run-time. Code re-generation from the higher abstraction model can become impractical since the context information is added manually. This is because developers need to apply some modifications, which often take place in the source code rather than in the abstraction model. Considering that the generation of adaptive UIs for new context of use requires developer's interventions, it makes it quite hard to provide dynamic context-aware UI adaptations at run-time. The integration of adaptive UIs in a running application presents further complexity that needs to be considered to meet the dynamicity of the current context of use.

In addition to the lack of methods for generating and integrating adaptive UIs at run-time of the application, there is also the problem of the development of usable UIs. In particular, building usable UIs is a tedious and a time consuming task. Many designers and developers lack the time and the expertise for providing highly usable UIs. As a result, the developed applications can have some usability issues. Additionally, the usability issues is caused by a lack of good design and best practices. One way to overcome these issues is to reuse expert knowledge about UI design. In this sense, design patterns can be considered in the applications' development process to capture expert knowledge. The concept of design patterns has been introduced as a way to share the design solutions of experienced developers [Gamma et al., 1993]. Moreover, design patterns have been presented as one step towards developing more usable systems at a faster rate than would be the case starting from scratch [Mahemoff and Johnston, 1998]. Nevertheless, choosing appropriate design patterns to be applied presents a burden due to the ambiguity of existing design solutions. There is currently no standardized method that states how to select HCI design patterns for the design of UIs. Besides, the availability of design patterns provides a novel opportunity for automated methods to support the reuse of existing design solutions. Few efforts, however, have been focused to investigate the use of design patterns for the development of adaptive UIs. For this reason, the selection of appropriate design solutions is usually elaborated manually without a well-defined process. Thus, the non-automatic selection of the required design patterns for designing the UI is an error-prone and time-consuming task. These two risk factors, including the error and time, should be avoided, especially in a dynamic context of use, where time costs and mistakes in the UI implementation directly affect user's satisfaction.

In summary, the above discussions reveal that some problems still need to be addressed. In particular, the main problems that this thesis focuses on are framed on the following points:

- Solve the heterogeneity between representations of design patterns by creating

models that can be used for selecting the relevant design patterns in order to enable the reuse of relevant design solutions.

- Improve the design and generation of UIs by using appropriate design patterns in order to help multidisciplinary teams developing usable interfaces.

1.3 Objectives and Research Questions

In order to address the critical issues discussed in the motivation and problem statement sections, this thesis proposes the reuse of expert knowledge to deal with the development of Web and mobile applications. Hence, the underlying objective of this thesis is to provide an approach for the design and generation of UIs using expert knowledge. We consider the use of HCI design patterns to capture expert knowledge in ways that it can be reused to solve design problems. The research goals derived from the underlying objective are the following:

- **RG-1:** The representation of knowledge related to HCI design patterns and their selection for UI specifications. This research goal can be split as follows:
 - **RG-1-1:** Define a model to represent design patterns that provide relevant design solutions for the development of UIs. The model developed in this thesis should allow the integration of knowledge regarding design patterns provided in the HCI domain and presented in existing catalogs.
 - **RG-1-2:** Propose a mechanism for the automatic selection of design patterns. This mechanism must be able to retrieve relevant HCI design patterns.
- **RG-2:** The automatic adaptation and generation of UIs at run-time. This research goal can be divided as follows:
 - **RG-2-1:** Define a mechanism for the dynamic generation of adaptive UIs. This mechanism has to integrate the selected design patterns to generate adaptive UIs that meet the user's need and current context.
 - **RG-2-2:** Establish techniques to integrate the generated UI code in an execution environment in order to allow the generation of UIs at run-time and to enable automatic UI adaptations to dynamic context changes.

In order to undertake these objectives, this thesis deals with the following research questions:

- **RQ-1:** How to solve the heterogeneity between representations of design patterns?
 - **RQ-1-1:** Which modeling methods can be used for representing design patterns?
 - **RQ-1-2:** Does the model capture and represent correctly design patterns?
 - **RQ-1-3:** How effective the developed model is in the context of design pattern selection?
 - **RQ-1-4:** Do the selected design patterns comply with the given design problems?

RQ-1 raises four sub-questions that concern the representation and selection of design patterns. Existing design patterns that are found in different sources with different representations need to be modeled following a methodological method, hence the RQ-1-1. The RQ-1-2 and RQ-1-3 arise from the need to build a correct and effective model. The RQ-1-4 emerges from the need to select relevant design patterns.

- **RQ-2:** How are UIs adapted and generated automatically at run-time? This research question refers to RG2. In order to answer this question, the following sub-questions must be addressed:
 - **RQ-2-1:** How can the selected design patterns be integrated with the design and generation of adaptive UIs to hasten the development process ?
 - **RQ-2-2:** Does the generation of UIs using design patterns lead to the development of usable interfaces?

RQ-2 entails two sub-questions that are related to the automatic generation of adaptive UIs and their integration in a running application. The selected design patterns that provide relevant design solutions need to be reused for the design and generation of adaptive UIs in order to improve the development process, hence the RQ-2-1. The RQ-2-2 emerges from the need to generate usable interfaces that meet users' requirements.

1.4 Thesis Contribution

The fundamental contributions of this thesis are inferred from the stated research questions. Specifically, the present thesis provides the following contributions:

- A framework for designing and generating adaptive UIs using expert knowledge. We design a framework that provides support for designers and developers to develop Web and mobile applications using HCI design patterns. The proposed framework is conceived as modular, open and accessible, and is characterized by code reuse.
- A modeling method to identify and build features that constitute the formal representation of design patterns. We propose the use of ontologies as formal models to represent knowledge about HCI design patterns.
- A design pattern recommender system to retrieve appropriate HCI design patterns for the current context of use. We provide a recommender system that enables the automatic selection of relevant HCI design patterns that offer solutions for a given design problem.
- A UI generation system to allow the automatic generation of application source code. We propose a system that considers mechanisms for run-time generation of adaptive user interfaces.
- A tool support and a developed prototype for validating the contributions of the present thesis. First, we develop a tool that helps designers and developers in selecting relevant design patterns. Then, we implement a tool that supports the automatic generation of UI components required for run-time adaptations.

1.5 Research Methodology

1.5.1 Methodological framework

Design science [Simon, 1996, March and Smith, 1995, Hevner et al., 2004] is a methodology that ultimately aims at fostering the development of artifacts to solve practical problems of the environment. Specifically, it is a methodology that seeks to “create novel artifacts in the form of models, methods, and systems that support people in developing, using, and maintaining IT solutions” [Johannesson and Perjons, 2014]. Design science research tasks consist of three main tasks, including problem investigation, building, and evaluation [March and Smith, 1995, Wieringa, 2009]. Design science methodology hence is a problem-solving paradigm that focuses on creating and evaluating artifacts, which serve a specific human purpose and tackle urgent problems [Gregor and Hevner, 2013]. This thesis is based on the design science methodology proposed by Wieringa [Wieringa, 2009, Wieringa, 2014]. According to this methodology, engineering and research problems are solved by decomposing them into two types of sub-problems, namely engineering and research sub-problems. The method applied for solving these problems is to follow a regulative cycle that starts with the problem investigation task; follows with solution, design, and validation tasks; and ends at the evaluation task. The regulative cycle could be an engineering cycle or a research cycle according to the type of problems to be solved. The engineering cycle includes problem investigation, solution design, design validation, solution implementation, and implementation evaluation tasks. While the research cycle has research problem investigation, research design, design validation, research, and analysis of results tasks.

1.5.2 Methodology applied to this thesis

We have selected the methodology proposed by [Wieringa, 2009, Wieringa, 2014] to guide the present thesis for its focus on Information Systems (IS) and software engineering research projects. By undertaking the objectives of this thesis, we will create and develop a set of artifacts such as a framework of UI generation, a formal method for representing design patterns, and systems that will support designers and software developers to determine how their applications can be developed using expert knowledge. These are considered as IT artifacts, which can be used to address the problem presented in this thesis, and hence they can be developed using the selected design science research methodology. This proved to be a relevant and an adequate methodology to carry out this research. According to the selected methodology, we consider the conceptual framework presented in Figure 1.1, where engineering and research cycles can be observed.

Following the methodology, shown in Figure 1.1, four engineering cycles and three research cycles are defined within the present thesis. A detailed description of each cycle is next introduced:

The first one is an engineering cycle (EC1) that aims at providing a framework for the design and generation of UIs using expert knowledge. First, we start investigating the problem (T1.1) by defining PhD motivations and goals. The problem investigation is done through studies of the literature, as well as systematic review surveys. After stating the goal and hypotheses of the research, we review and analyze the state of the art in UI design and specification (T2.1). The review of the state of the art reveals that there are few studies that consider the use of expert knowledge for the development of UIs. Therefore,

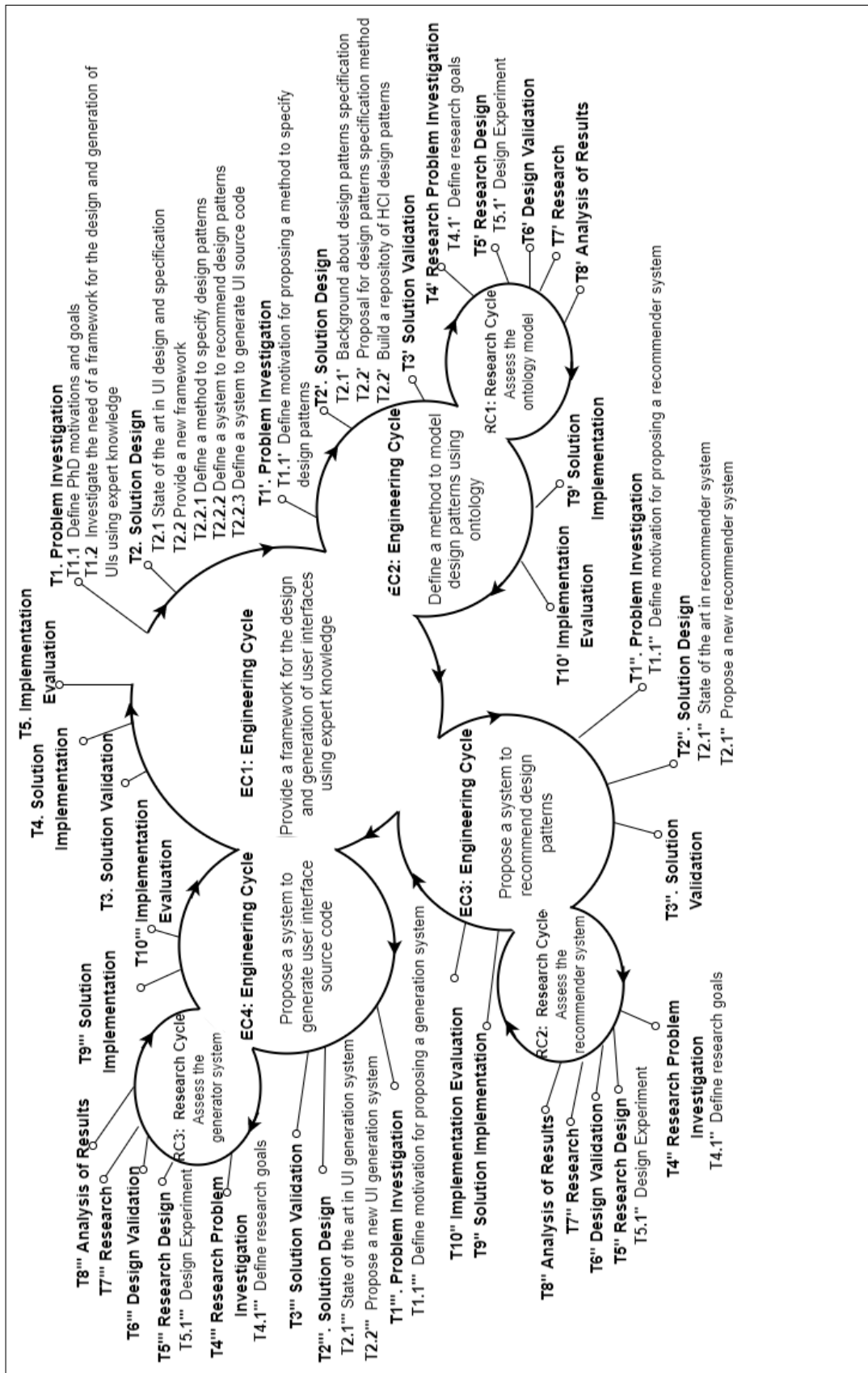


Figure 1.1: Research methodology overview of this thesis

we propose a set of artifacts to allow the integration of design patterns into the UI design and generation framework. To this end, we first define a method to model design patterns using ontologies (T2.2.1), then we propose a system to recommend HCI design patterns (T2.2.2), finally we provide a system to generate UI source code (T2.2.3).

In order to perform the definition of a method to model design patterns (T2.2.1), we propose a second engineering cycle (EC2). This cycle starts with problem investigation (T1.1'). For problem investigation, studies of the literature and systematic review surveys are carried out. Then, the solution design involves review of background about design patterns specifications (T2.1'). The purpose of this review is to investigate existing specification methods and to highlight the need to propose a new method. Next, we propose a method to specify design patterns in the HCI domain (T2.2'), and a repository of HCI design patterns (T2.3'). After that, the ontology model provided by the proposed specification method is validated (T3'). In order to conduct the validation, a first research cycle (RC1), that includes tasks T4' to T8', is required. For validating the ontology model, we implement a tool that supports the proposed method (T9'). Finally, the implementation is assessed regarding specific evaluation criteria (T10').

Once the second engineering cycle (EC2) ends, a third engineering cycle (EC3) is proposed to define a system for recommending HCI design patterns. EC3 starts with problem investigation (T1.1''). Then, the solution design requires reviewing and analyzing existing recommender systems (T2.1''), which is done by systematic review surveys about design pattern recommendation. This review shows that there are no previous recommender systems that integrate ontology models to recommend HCI design patterns. Subsequently, we propose a new system for recommending HCI design patterns using ontology models (T2.1''). Next, the defined recommender system is validated. To perform this validation, a second research cycle (RC2), that includes tasks T4'' to T8'', is required. After validating the recommender system, we implement a prototype that supports the recommender system (T9''). Finally, the implementation is assessed considering specific evaluation criteria (T10'').

Once the third engineering cycle (EC3) ends, another engineering cycle (EC4) is proposed. The purpose of EC4 is to define a system for generating the UI source code. This cycle starts with problem investigation (T1.1'''). In particular, studies of the literature, as well as systematic review surveys are carried out to perform the problem investigation. This task is followed by the solution design that requires analyzing state of the art in UI generation systems (T2.1'''). This review reveals that there are no previous works that automatically generate adaptive UI source code using design patterns and ontology models. Subsequently, we propose a new generator system for generating UIs' source code (T2.1'''). Next, the defined generation system is validated through a third research cycle (RC3). This research cycle includes tasks T4''' to T8'''. After validating the generation system, we implement a tool that supports the generation of UI source code (T9'''). Finally, the implementation is assessed regarding specific criteria (T10''').

After EC4, it is possible to integrate expert knowledge; more specifically design patterns, within the proposed framework for the design and generation of UIs. After that, the proposed framework is validated (T3), implemented (T4), and evaluated (T5).

1.6 Thesis Outline

The structure of the present thesis is organized in nine chapters, which are part of the research tasks identified in the methodology outlined in Figure 1.1. First, the problem investigation, which is performed in Tasks T1, T1', T1'' and T1''', is described in Chapters 1, 2 and 3. Then, we discuss the solution design, for the engineering cycles EC1 through EC4 and the research cycles RC1 through RC3, in the following Chapters: Chapter 4, 5, 6, and 7. Finally, the solution implementation and solution validation are described in Chapter 8.

The contents of the remainder of this thesis are described below:

- **Chapter 2 - Background:** This chapter summarizes and explains the most relevant background concepts related to this thesis in order to provide the reader with the fundamentals for the overall thesis work. It covers a description of the main fields, which are related to the work presented in this dissertation.
- **Chapter 3 - State of the Art:** This chapter includes a review of the state of art associated with the thesis topic. This review is centered on analyzing the literature to highlight relevant related works.
- **Chapter 4 - AUIDP: A Framework for the Design and Generation of User Interfaces:** This chapter presents the framework, named AUIDP, which is developed for the design and generation of UIs using expert knowledge. First, we provide an overview of the main building blocks that constitute the proposed framework and then, we go on describing the framework from an architectural point of view, and presenting the framework implementation details.
- **Chapter 5 - Design Pattern Specification Method:** This chapter introduces the method that focuses on the formal specification of HCI design patterns by means of ontologies. This method is based on a methodological approach for building ontology models. This chapter also describes the created MIDEP ontology that represents knowledge about HCI design patterns.
- **Chapter 6 - Design Pattern Recommender System:** This chapter describes the IDEPAR system that focuses on recommending the most relevant HCI design patterns for a given design problem. In this chapter, we provide an overview of the proposed recommender system, along with a description of its main modules, and a description of its implementation.
- **Chapter 7 - User Interface Generator System:** This chapter presents a description of the second system within the global framework. The present system, named ICGDEP, is provided for the generation of the final UIs' source code. An overview of the ICGDEP system, along with a description of its main modules and the developed implementation are given in this chapter.
- **Chapter 8 - Evaluation:** This chapter describes how the proposed solution has been evaluated by means of several experiments. In this chapter, we present a detailed description of the validation process highlighting the use of empirical research cycles to formally validate the contributions of this thesis.

- **Chapter 9 - Conclusions and Future Works:** This chapter summarizes the main contributions of the present thesis and highlights future research directions related to the work proposed. Additionally, this chapter incorporates the publications generated during this thesis development.

Background

2.1 Introduction

The present thesis relies on different concepts to deal with the development of an approach to include expert knowledge for the design and generation of UIs. These concepts are introduced in this chapter in order to clarify the foundations in which this work relies and to present the background information required to grasp the research area of this thesis.

The present chapter gives an overview of concepts and research in the field that are relevant to this thesis. Specifically, the remainder of this chapter is organized as follows: Section 2.2 discusses fundamentals of design patterns. Section 2.3 provides an overview about ontologies. Finally, the summary of the present chapter is included in Section 2.4.

2.2 Fundamentals of Design Patterns

This section provides fundamentals about design patterns. First, the design pattern concept is introduced and subsequently other relevant notions are presented, together with some background on the domain and documentation styles of design patterns. Finally, existing design pattern specification methods are described.

2.2.1 Design pattern concept

The design pattern concept was originally introduced by Alexander et al. into the field of architecture in late 1970s [Alexander et al., 1977]. Alexander et al. created the design pattern concept to deal with problems occurring in building architecture and to enhance the design process of building and urban areas. According to Alexander, a design pattern is considered as "a careful description of a perennial solution to a recurring problem within a building context, describing one of the configurations that brings life to a building". In addition, Alexander developed a theory to describe design patterns. The core of Alexander's theory is to identify the structure of design patterns. In Alexander's theory, "each pattern describes a problem that occurs over and over again in our environment, and then describes the core solution to that problem, in such a way that you can use the solution a million times over, without ever doing it the same way twice".

Later, in Alexander's view [Alexander et al., 1979], a design pattern is characterized as "a three-part rule, which expresses a relation between a certain context, a problem, and a solution". A design pattern is, thus, a solution to a recurring problem in a given context, as illustrated in Figure 2.1. In other words, the context is the design situation that gives rise to a problem, the problem is a set of influencing factors or forces that repeatedly arise in the context, and the solution is the configuration that solves the problem.

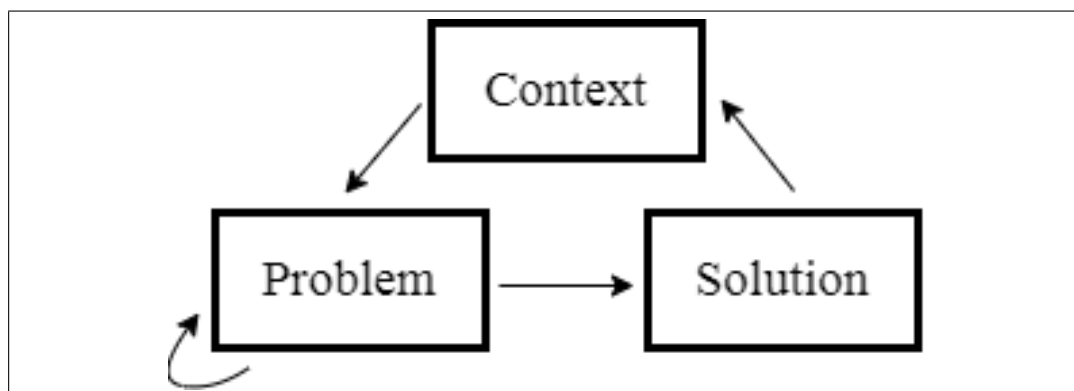


Figure 2.1: Design pattern structure

Furthermore, according to Gabriel [Gabriel, 1996], a design pattern is considered as "a constituent of expression, repetition, and configuration, in which expression is about a certain context that is repeatable but can also be resolved through a specific system configuration". In turn, according to the GoF book, Gamma et al. [Gamma et al., 1995] defined a design pattern as a description of communication objects and classes that are customized to solve design problems within a particular context.

2.2.2 Design pattern domain

Design patterns have been proposed in many domains to provide a reusable form of a solution to a design problem. It began in the architecture and civil engineering field [Alexander et al., 1977]. In this field, Alexander introduced an initial set of 253 design patterns that entailed the design of buildings and architectural plans. In the early 1990s, the design pattern concept was transferred to the software engineering field and applied to object programming. It became popular in this field after the presentations in OOPSLA 94 [Beck, 1987] and the publication of the GoF book [Gamma et al., 1995]. The GoF book, published by [Gamma et al., 1995], described 23 object-oriented design patterns that have greatly influenced current programming languages and are still regarded as an important source to address the reusability of software components.

Later, the design pattern concept was adopted in the HCI domain to capture HCI knowledge [Borchers, 2008, Kruschitz and Hitz, 2009]. The HCI community has intended to tackle the quality of HCI design by promoting design patterns. In this context, HCI design patterns, also called UI design patterns and interaction design patterns, have been introduced to help the identification of concrete design elements that provide solutions to interaction design problems [Seffah, 2015a, Seffah, 2015b]. Thereby, many different collections of HCI design patterns have been published by HCI experts over the last few years. Among the heterogeneous collections of HCI design patterns, the "Common Ground"

[[Tidwell, 1999](#)], "Experience" [[Coram and Lee, 1996](#)], and "Amsterdam" [[Welie, 1999](#)] catalogs are very popular and widely used in the HCI domain.

2.2.3 Design pattern documentation

There is no single well-defined or standard format for documenting design patterns. Rather, a variety of different formats has been introduced by various design pattern authors. Within the original documentation, which is provided by [[Alexander et al., 1979](#)], design patterns were described using natural language and defined using a subset of elements comprising the design pattern name, problem, usage, solution, and examples.

All the existing forms of design pattern documentation include the basic characteristics of each design pattern. Table 2.1 gives a list of the formats proposed by different authors for documenting design patterns.

Table 2.1: Design pattern documentation forms

Documentation	Design Pattern Elements
Alexander Form ¹	Name, Problem, Usage, Solution, Examples.
GoF Form ²	Name, Alias, Problem, Context, Forces, Solution, Example, Resulting context, Rational, Known uses, Related design patterns.
Thomas Form ³	Requirements, Icon, Summary, Problem, Solution, Application, Impact, Relationships, Case study Example.
Buschmann Form ⁴	Summary, Context, Problem, Solution, Structure, Dynamics, Variants, Example resolved, Implementation, Consequences, Known uses.
Bishop Form ⁵	Role, Design, Implementation, Illustration, Example, Use, Exercises.
Vora Form ⁶	Name, Problem, Solution, How, Why, Related design patterns.

^a[[Alexander et al., 1979](#)]

^b[[Gamma et al., 1995](#)]

^c[[Erl, 2007](#)]

^d[[Buschmann et al., 2007](#)]

^e[[Bishop, 2007](#)]

^f[[Vora, 2009](#)]

2.2.4 Design pattern language

Alexander define a pattern language as real world elements that describe good design practices in order to simplify and solve the problem [[Alexander et al., 1979](#)]. In Coplien's definition, "A pattern language is a structured collection of patterns that build on each other to transform needs and constraints into an architecture. It is not a programming language in any ordinary sense of the term, but is a prose document whose purpose is to guide and inform the designer" [[Coplien, .](#)]. According to [[Kruschitz and Hitz, 2010](#)], a pattern language is characterized as " complete set of patterns for a given family of design problems in a given domain. A pattern language describes problems by means of high-level design patterns, which may be solved by lower-level design pattern".

In the context of the CHI '2003 conference, a workshop on HCI patterns was held in order to provide the definition of a pattern language to express HCI design patterns. In this sense, a first version of Pattern Language Markup Language (PLML) is developed

by [Fincher et al., 2003]. Particularly, PLML version 1.1 is an XML-based language for writing HCI design patterns. It comprises different description elements to standardize the definition of design patterns.

An enhanced version of PLML, called eXtended Pattern Language Markup Language (XPLML), is introduced by [Kruschitz, 2009]. This extension includes some attributes that are not presented in the PLML language in order to define and specify a unified HCI pattern form. XPLML is based on XML to allow the specification of content elements, relations and metadata of HCI design patterns.

In order to overcome the weakness of PLML and its extensions, [Thanh-Diane et al., 2016] develop an XML-compliant markup language named User Interface Pattern Language Markup Language (UIPLML). UIPLML aims at defining UI design patterns for various contexts of use in order to provide information, which are required to design a UI.

2.2.5 Design pattern specification

The specification and description of design patterns are required for their successful implementation and recovery. Design patterns can be expressed informally, semi-formally, or formally.

Design patterns have traditionally been specified informally using textual descriptions and informal diagrams. The documentation of design patterns in existing catalogs and collections is mostly presented in non-formal representation [Di Martino and Esposito, 2016]. Such representation is useful for communications among designers or developers. Nevertheless, it is usually difficult to perform the selection of appropriate design patterns using informal descriptions, which is essential for describing properties of some design patterns, since they are descriptive and they lack support of abstraction. In addition, informal specification is often ambiguous, imprecise and inadequate for automated processing [France et al., 2004].

Along with informal specification of design patterns, other semi-formal representations have been provided in different catalogs, especially to reduce the complexity and to enhance human understanding of patterns. The majority of these representations are supported by graphical notations like UML diagrams [Sunyé et al., 2000]. Semi-formal specification is helpful to a human reader, since it supports all information required to understand a design pattern. The major problem of this type of specification is that the content of design patterns, which are informally represented, could not be automatically analyzed.

While informal and semi-formal specifications are useful for understanding a design pattern and guiding through its description, they only provide the structural aspects of the design pattern. These representations do little to help designers and developers understand the higher-level concern of each design pattern. To allow better understanding of design patterns and to achieve automated support for pattern-based development, it is necessary to use formal specifications using ontologies.

2.3 Overview of Ontologies

This section gives an overview about ontologies, their different standards and methodologies to describe and design them. The remainder of this section is structured as follows:

Subsection 2.3.1 introduces the definition of an ontology. Subsection 2.3.2 describes the components that characterize an ontology. Subsection 2.3.3 presents the existing methodologies used to design and create ontologies. Subsection 2.3.4 introduces ontology development tools. Finally, Subsection 2.3.5 describes existing ontology evaluation methods.

2.3.1 Ontology definition

Originally, the concept of ontology has stemmed from philosophy, where it is concerned with the study of existence. Then, this term was used intensively in the computer science and artificial intelligence fields to support the reuse of knowledge that is formally represented. In the last decade, various definitions of the ontology concept have been provided. For instance, a common definition of ontology has been previously introduced by [Gruber, 1993, Gruber, 1995]. The latter considered as an explicit specification of a conceptualization, where “explicit” refers to the specification of concepts in a specific domain of knowledge, whereas “conceptualization” consists of the abstraction of a domain of interest. This definition has been further merged by Studer et al. [Studer et al., 1998] stating that “an ontology is a formal, explicit specification of a shared conceptualization”. More explicitly, an ontology is defined as “a hierarchically structured set of concepts describing a specific domain of knowledge that can be used to create a knowledge base” [Blomqvist and Sandkuhl, 2005]. During the past few years, ontologies have been evolved and largely used in various domains. Nowadays, ontologies are becoming extremely important and widely adopted in many fields such as Semantic Web, ISs, or knowledge management, where they play a crucial role in organizing and representing knowledge.

Currently, ontologies are viewed as a formal knowledge representation that is based on formal specification languages and modeling tools to build knowledge models, so that knowledge can be represented in a machine-readable way. This contributes to a better understanding and enables the analysis of knowledge in a specific domain. The effective use of ontologies requires a well-designed development language. Over the past few years, numerous ontology development languages have been proposed. Most of these languages are based on the eXtensible Markup Language (XML). Notable examples of existing standards are the following: OIL [Fensel et al., 2001], DAML-S [Martin et al., 2002], Web Service Modeling Language (WSML) [de Bruijn et al., 2004], Resource Description Framework (RDF) and RDF Schema (RDFS) [Klyne, 2004], and the Ontology Web language (OWL) [Patel-Schneider, 2004].

2.3.2 Ontology components

There exist various components that characterize an ontology [Slimani, 2015] as shown in Figure 2.2. The main components of an ontology are classes, individuals, relations, attributes, functions, axioms, and restrictions. A description of each of these components is provided below.

- **Classes:** This component, also known as concepts or type of objects, describes the ontology concepts or tasks that are generally organized in taxonomies.
- **Individuals:** This component, also known as instances, represents the specific elements of the ontology classes.

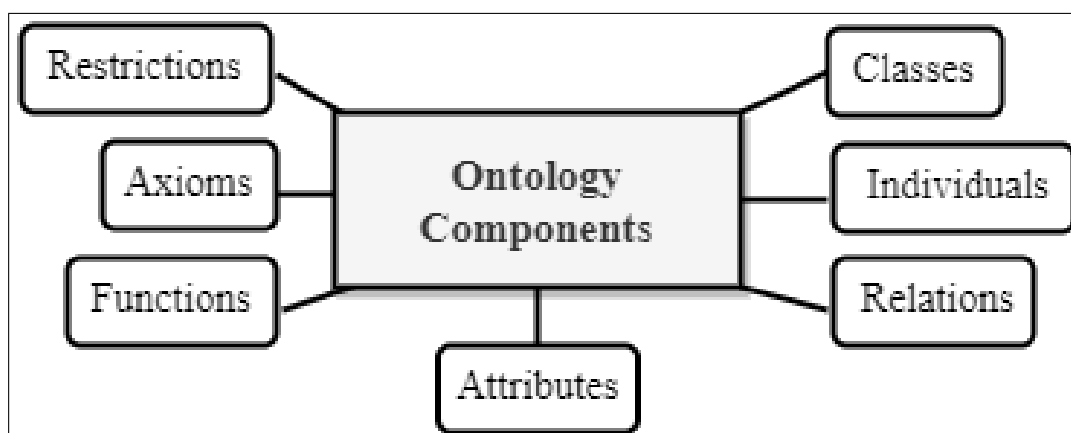


Figure 2.2: Overview of ontology components

- Relations: This component represents the interaction between different ontology concepts.
- Attributes: This component, also known as properties or features, represents the properties of the ontology classes.
- Functions: This component is considered as a complex structure that is formed from specific relations.
- Axioms: This component represents the hierarchies of concepts and the relations among them. It can be classified as consolidation, epistemological, and derivation axioms.
- Restriction: This component is used to explicitly define the ontology constraints.

2.3.3 Methodologies for designing ontologies

Recently, the popularity of ontologies has exploded, mainly due to the widespread interest on the Semantic Web. As a result, various methodologies for building ontologies are emerging. All existing methodologies have yielded major progress, transforming the art of designing ontologies to an engineering activity. These methodologies provide a set of principles, processes, and methods to serve as a guide for building, constructing, and evaluating ontologies.

In the literature, a variety of different methodologies for ontology engineering have been proposed. In this subsection, several well-known methodologies to follow, including METHONTOLOGY [López et al., 1999], Ontology Development 101 [Noy et al., 2001], DILIGENT [Pinto et al., 2004], On-To-Knowledge [Sure et al., 2004], and NeOn methodology [Suárez-Figueroa et al., 2012], are described. These methodologies among others have gone a step forward towards transforming the art of building ontologies to an engineering activity. Within all these methodologies, requirements engineering represents a crucial role for designing ontologies. In this context, competency questions are generally used to formalize ontology requirements. They are defined as a set of problems that should be solved by the logic axioms of ontologies [Grüniger and Fox, 1995].

2.3.3.1 METHONTOLOGY

METHONTOLOGY [López et al., 1999] is a methodology for building ontologies, at the knowledge level, either from scratch or by reusing existing ontologies. Its ontology building process comprises the identification of the ontology development process, a life cycle, and some techniques and support activities. This methodology proposes the following concrete development activity steps.

- **Specification:** In this step, the ontology's purpose, scope, level of formality, and other information are identified. This step will make an ontology specification document.
- **Knowledge acquisition:** This step aims to gather knowledge needed to understand the domain and concepts of the ontology.
- **Conceptualization:** In this step, the representation of the domain knowledge in a conceptual model is performed in order to identify the ontology concepts, relations, instances, and properties.
- **Integration:** The goal of this step is to reuse existing ontologies by incorporating concepts from other ontologies in order to speed up the development process.
- **Implementation:** In this step, the ontology is represented in a formal language using an implementation tool.
- **Evaluation:** Within this step, the ontology is validated and verified.
- **Documentation.** This step aims to create a document that includes all information regarding the ontology development life cycles.

2.3.3.2 Ontology Development 101

Ontology Development 101 [Noy et al., 2001] is a methodology that presents a structured overview of the tasks that are required to build an ontology. This methodology provides an iterative process that includes the following steps:

- Define the ontology domain and scope using the competency questions.
- Reuse existing ontologies.
- Enumerate the relevant terms that could be used
- Define the ontology classes, the class hierarchy, and the ontology properties.
- Create the ontology instances.

2.3.3.3 DILIGENT

DILIGENT [Pinto et al., 2004] is a methodology that is intended to support its participants in order to collaboratively build one shared ontology. This methodology is focused on collaborative ontology engineering allowing the creation of different ontology versions. Its development process is divided into five main phases, including build, local adaptation, analysis, revision, and local update. In the following, we provide a description of these phases.

- **Build:** In this phase an initial ontology is created by a team of domain experts, users, knowledge engineers and ontology engineers.
- **Local adaptation:** In this phase, ontology users are involved in updating the initial ontology according to their needs.
- **Analysis:** Within this stage, the local variants of the developed ontology are analyzed.
- **Revision:** In this phase, the ontology is updated according to the previous phase and a new version is released.
- **Local update:** In this phase, users may align their local ontologies with the new version.

2.3.3.4 On-To-Knowledge

On-To-Knowledge [Sure et al., 2004] is a methodology that aims to build ontologies for knowledge management systems. This methodology consists of five phases: kick-off, refinement, evaluation, and application and evolution. These phases are described below.

- **Kick-off:** This phase requires the collaboration between domain experts and ontology engineers in order to create the Ontology Specification Requirement Document (ORS) and to develop an initial semi-formal model.
- **Refinement:** In this phase, ontology engineers are involved to formalize the initial model, which is derived from the previous phase, into a real ontology.
- **Evaluation:** This phase aims to evaluate the knowledge-based system in which the ontology is considered.
- **Application and evolution:** In this phase the knowledge-based system is deployed.

2.3.3.5 NeOn

NeOn [Suárez-Figueroa et al., 2012] is a scenario-based methodology that provides the definition and formalization of ontology. It is defined as: "A set of nine flexible scenarios for collaboratively building ontologies and ontology networks, placing special emphasis on reusing and re-engineering knowledge resources (ontological and non-ontological)" [Suárez-Figueroa et al., 2012]. The NeOn methodology defines a set of nine scenarios, which emphasizes its flexibility and adaptability to various development scenarios. These scenarios are divided into activities and can be combined in a reusable way. In particular, the NeOn methodology provides the following:

- **A glossary of terms:** This glossary is required for the ontology engineering processes and activities.
- **A set of nine scenarios:** These scenarios describe how the ontology is built in the different following ways:
 - Scenario 1: From specification to implementation.
 - Scenario 2: Reusing and re-engineering non-ontological resources.

- Scenario 3: Reusing ontological resources.
 - Scenario 4: Reusing and re-engineering ontological resources.
 - Scenario 5: Reusing and merging ontological resources.
 - Scenario 6: Reusing, merging and re-engineering ontological resources.
 - Scenario 7: Reusing ontology design patterns.
 - Scenario 8: Restructuring ontological resources.
 - Scenario 9: Localizing ontological resources.
- Two life cycle models: These life cycles allow describing how the ontology development processes and activities could be organized into the project phases.

2.3.4 Ontology development tools

Over the years, a considerable number of ontology development tools, which provide an environment for developing and manipulating ontologies, have been proposed. Most of these tools are considered as ontology editors and ontology visualization tool. The commonly used tools are presented as follows:

- Ontolingua [Farquhar et al., 1997]: It is an ontology tool that includes a set of services to provide users the ability to create, browse, edit, and publish ontologies.
- OILED [Bechhofer et al., 2001]: It is an ontology editor that facilitates the development of ontologies using reasoning to support the ontology design.
- UBOT [Kogut et al., 2002]: It is a UML-based tool for building ontologies by creating a prototype UML profile for DARPA Agent Markup Language.
- OntoEdit [Sure et al., 2002]: It is an ontology editor that integrates various ontology engineering aspects. It provides several interfaces to create and edit ontology entities. This tool supports different ontology languages, including OWL, RDF, and RDFS.
- Protégé [Gennari et al., 2003]: It is an open source ontology development tool that provide a graphic UI for creating ontologies. This tool allows generating ontology files in different formats such as RDF-XML or OWL-XML. The significant advantage of the Protégé tool is its ability to create and process large ontologies in an efficient way.
- Swoop [Kalyanpur et al., 2006]: It is a semantic Web ontology tool editor that allows creating and editing ontologies. It supports the OWL ontology language, includes an OWL validation and provides different OWL presentation syntax views.

2.3.5 Ontology evaluation

Like all software artifacts, it is mandatory to assess and validate ontologies. According to [Gómez-Pérez, 2004], ontology evaluation is introduced in the view shown in Figure 2.3, where it is about finding the distance between the model, which is the approximate conceptualization, and the real world.

Subsequent subsections discuss the ontology evaluation as provided by literature. First, Subsection 2.3.5.1 introduces different evaluation aspects of an ontology. Then, Subsection 2.3.5.2 describes several ontology evaluation methods. Finally, ontology evaluation criteria are defined in Subsection 2.3.5.3.

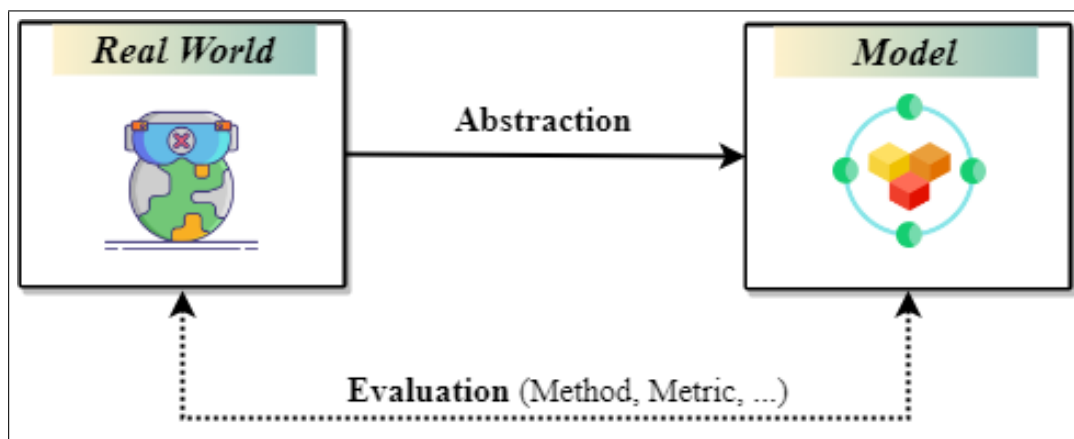


Figure 2.3: Ontology evaluation based on the definition of ontologies

2.3.5.1 Ontology aspects

Ontologies can be evaluated under different aspects. These aspects are discussed by [Vrandečić, 2009] and summarized as follows:

- **Vocabulary:** This aspect is about all names presented in the ontology. It can be either the ontology URIs or the ontology literals. The URI represents the signature of the ontology, whereas literals are a tuple with a literal value and a URI that identifies the datatype of the literal value.
- **Syntax:** This aspect deals with the different syntaxes that describe an ontology. Common syntaxes that allow serializing ontologies are RDF/XML [Beckett and McBride, 2004] and OWL Abstract Syntax [Patel-Schneider, 2004].
- **Structure:** This aspect concerns the RDF graph described in the ontology.
- **Semantics:** This aspect captures the semantic features that are defined as the common characteristics of an ontology.
- **Representation:** This aspect deals with the relation between the ontology's structure and semantics in order to uncover mistakes between the formal specification and the shared conceptualization of the ontology.
- **Context:** This aspect is about the ontology features that describe artifacts accompanying the ontology, including the data source described in the ontology or the application that uses the ontology.

2.3.5.2 Ontology evaluation methods

In the last decade, several ontology evaluation methods for validating ontologies have been proposed by researchers. The different known evaluation methods can be mainly assigned

to multiple kinds of methods [Raad and Cruz, 2015, Hu and Liu, 2020]. We describe the most commonly used methods as follows:

- **Competency question evaluation method:** This evaluation method, is considered as one of the earliest methods towards ontology evaluations [Grüniger and Fox, 1995]. It focuses on evaluating the ontology with regards to competency questions, which represents the questions that an ontology should answer. In this method, competency questions need to be formalized in a query language and integrated in a tool to allow an automatic competency question evaluation.
- **Technology-based method:** This method focuses on ontology evaluation by using tools to investigate the characteristics of ontologies. In the following, we present a brief description of existing tools that support ontology evaluation, including OntoMetric [Lantow, 2016], OntoClean [Fernández-López and Gómez-Pérez, 2002], OntoManager [Hameed et al., 2002], and OOPS! [Poveda-Villalón et al., 2012].
 - OntoMetric [Lantow, 2016]: It is an online tool that provides a Web-based platform for evaluating ontologies by calculating the following five metrics: (i) Base metric refers to the quantity of ontology elements, including ontology, class, individuals, properties. (ii) Schema metric concerns the ontology design, such as the richness of attributes that are related to the number of attributes created for each ontology class. (iii) Class metric refers to the connectivity and fullness of ontologies' classes. (iv) graph metric is related to depth and cardinality of ontologies' structures. (v) Knowledge-base metric is about the distributions of instances across ontologies classes.
 - OntoClean [Fernández-López and Gómez-Pérez, 2002]: This tool is provided as a plugin of WebODE that is based on the OntoClean methodology proposed by [Guarino and Welty, 2002]. The aims of this methodology is to provide a formal evaluation of the taxonomy structure. Consequently, the present tool allows evaluating formal ontologies to detect inconsistencies by (i) cleaning the taxonomical structure of ontologies, such as the upper level of ontologies, and (ii) comparing ontologies with predefined taxonomical structure.
 - OntoManager [Hameed et al., 2002]: This tool is used by administrators, domain experts and business analysts to identify the truthfulness of ontologies regarding its problem domain. It is an ontology-based information portal that helps in discovering changes occurred in the ontology, finding the “weak places” in the ontology and modifying the ontology to improve it regarding end users requirements.
 - OOPS! [Poveda-Villalón et al., 2012]: It is a Web-based tool that acts as an ontology pitfall scanner to evaluate ontologies against a set of 41 potential errors called pitfalls. These pitfalls are identified using OPPS! by analyzing ontologies and extracting the detected pitfalls that could lead to some modeling errors. Each pitfall has a specific level that indicates its importance, namely critical, important, minor. First, critical means that it is crucial to correct this type of pitfall since it could affect the ontology. Second, important indicates that the pitfall is not critical for ontology function but it is important to correct it. Finally, minor reveals that this type of pitfall is not a problem.

- **Gold standard-based method:** This method aims to evaluate the ontology against a gold standard, which serves as a reference [Zavitsanos et al., 2010]. This gold standard is considered as a set of strings that provides a representation of the concepts of a problem domain. In this sense, the gold standard could be either another ontology or a corpus of documents that are created by domain experts.
- **Application-based method:** This method, also named task-based evaluation, allows checking how well an ontology could support a specific application or task. It attempts to assess the applicability and usability of an ontology by proving the performance of an ontology on a specific task [Porzel and Malaka, 2004]. In general, this evaluation method consists of measuring how effective an ontology is in the context of real applications. It allows evaluating ontologies according to the expected outputs of an application or its performance on the given tasks. The evaluation can be done using the feedback of users to check whether the ontology successfully support specific functionalities within an application.
- **Criteria-based method:** This method, known as quality-based evaluation method, allows measuring how far an ontology adheres to specific evaluation criteria. It allows measuring the individual quality attributes of an ontology using the pre-defined criteria by introducing a numerical score [Pak and Zhou, 2009]. To calculate these measures, two approaches are defined: a “structure-based” approach, and a “complex and expert-based” approach. First, the structure-based approach focuses on evaluating a given taxonomy by calculating different structure properties. Second, the complex and expert-based approach aims at evaluating ontologies by using ontology quality aspects.

2.3.5.3 Ontology evaluation criteria

The evaluation of ontologies can target a number of various criteria. In this regard, several criteria have been introduced to evaluate the ontology. The criteria, discussed in the literature [Gómez-Pérez, 2004, Gangemi et al., 2005], are summarized as follows:

- **Accuracy:** This criterion determines the correctness of an ontology by checking if the ontology correctly represents the real world. A higher accuracy is obtained from correct definitions and descriptions of the ontology’s classes, properties, and individuals.
- **Completeness or competency:** This criterion checks if all relevant information is appropriately covered in the ontology. It states if the ontology could answer the defined competency questions.
- **Conciseness:** This criterion is useful to check if an ontology includes redundant semantic representations or irrelevant information with regard to the domain to be covered by the ontology.
- **Adaptability:** This criterion measures how far an ontology anticipates its uses. It determines the ease of use of an ontology in multiple contexts by verifying the possibilities to extend and specialize the ontology without the need to remove axioms.

- **Clarity:** This criterion, also named sensitiveness, determines how effectively an ontology communicates the expected meaning of the defined terms.
- **Consistency or coherence:** This criterion reflects that the ontology does not contain or allow for any contradictions.
- **Computational efficiency:** This criterion determines the ability of tools to interact with the ontology. In particular, it refers to the speed that reasoners need to complete the required tasks.

2.4 Concluding Remarks

The background information, described in this chapter, represents the backbone of the present dissertation since it is related to the thesis scope. In particular, we have introduced some fundamentals of design patterns and provided an overview of ontologies with the aim of giving a better comprehension about the research area of this thesis.

The present thesis explores the use of HCI design patterns for the design and generation of UIs. In this sense, we focus on providing a formal specification of design patterns in the HCI domain. To this end, ontologies are considered to represent knowledge about HCI design patterns. First, we adopt the Neon methodology in order to build an ontology that includes knowledge about HCI design patterns. Second, we select OWL as an ontology development language. Then, we choose the Protégé tool in order to conduct the implementation of the proposed ontology. Finally, we follow a three-fold approach for evaluating the developed ontology according to three evaluation criteria, namely completeness, conciseness, and consistency. This evaluation approach relies on the use of the competency question evaluation, the technology-based, and the application based methods.

The background information defined in the present chapter are used in Chapter 5 and Chapter 8, where the design pattern specification method is introduced and the ontology evaluation method is described.

State of the Art

3.1 Introduction

Supporting the design and generation of UIs raises numerous challenges. These challenges involve both research needs regarding the development of adaptive applications, as well as perspectives related to the practical use of design patterns.

Throughout this chapter, we provide a comprehensive insight of the literature review that is relevant to the research reported in the current PhD thesis. The state of the art, presented in this chapter, is a fundamental part that helps to sharpen the contributions of this thesis and to position this work among research domains. The present thesis can be considered in the intersection of different research aspects as shown in Figure 3.1. For this thesis, four research aspects have been identified, including model-based development, mobile and Web development, adaptive UIs and design patterns. With regard to each of these aspects, relevant works have been carried out and described in this chapter.

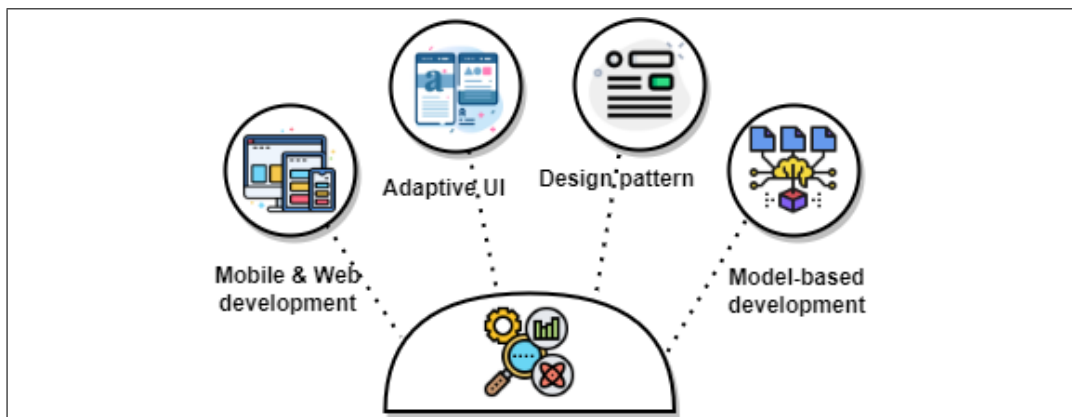


Figure 3.1: Research aspects

The remainder of this chapter is organized as follows: Section 3.2 is centered on analyzing current approaches for the specification of UIs, distinguishing the approaches that follow a pattern-based method from the other ones. Section 3.3 explores the existing

literature on systems focusing on the recommendation of design patterns to solve a given design problem. Section 3.4 describes the reviewed proposals about the adaptation and the generation of UIs. Finally, this chapter ends with some concluding remarks regarding the gaps in the current state of the art.

3.2 User Interface Specification Approaches

The aim of this section is to provide an overview of the current works that present relevant features for supporting the design and specification of UIs. In subsection 3.2.1, UI specification works that rely on model-based or pattern-based methods are described. In subsection 3.2.2, analysis and discussion of the surveyed works are addressed.

3.2.1 Review of user interface specification

Several methods and tools have been provided for supporting the specification of UIs. The ultimate goal of these methods is to provide ways for building and designing interface models. In this sense, UIs can be designed with model-based methods. Additionally, design patterns can implicitly be part of these UI specifications. Relevant proposals in this direction are described below.

3.2.1.1 Model-based specification

MDD is an approach for the design and development of a system and its UI independently of the supported platform [De Oliveira et al., 2013, Andemeskel and Semere, 2018]. Generally, in the context of Model-Based User Interface Development (MBUID), the UI can be specified with a high level of abstraction [Meixner et al., 2011]. The interest of these methods is focused on conceptual models and transformations between these models.

Calvary et al. [Calvary et al., 2003] develop the CAMELEON Reference Framework (CRF), which is a unified framework for UI model-based and MDD. The CRF allows designers to design multi target UIs. Within this framework, UIs are represented on four basic levels of abstraction starting from task and concept level through Final User Interface (FUI). The task and domain models list the hierarchies of tasks performed by users while interacting with the UI and define the information handled by the application. The next level is the Abstract User Interface (AUI) model that articulates the UI by means of abstract interaction objects. The third abstraction level is the Concrete User Interface (CUI) model and it describes the UI in terms of concrete interaction objects. The FUI layer is obtained through the translation of the concrete UI and represents the UI by implementation-dependent source code.

The UsiXML is developed by Limbourg et al. [Limbourg et al., 2004] to support the specification of UIs independently of a specific interaction technique or a certain computing platform. Particularly, it allows the specification of domain, task, abstract UI, concrete UI and context models. UsiXML stems from CRF and therefore it is structured according to four abstraction levels by allowing designers to specify the UI on tasks and concepts, AUI, CUI, and FUI levels defined by CRF. To achieve the UI design, UsiXML comprises five models, including task, domain, presentation, dialog, and context of use models. It is also based on a transformational approach to allow model transformations.

TERESA XML [Berti et al., 2004a] is defined as an integral part of the Transformation Environment for inteRactivE Systems representations (TERESA) [Berti et al., 2004b] that

is introduced for the design and the development of multi-device UIs. Its main target is to consider a so-called “one model, many interfaces” approach in order to allow model-based Graphical User Interface (GUI) development for various devices using the same Concur Task Tree (CTT) model that is stored in XML-compliant format. In general, TERESA XML serves as a solution for task modeling using CTT notation, as well as for abstract and concrete UI descriptions. Subsequently, TERESA XML covers two main abstraction levels, including the model level, the abstract, and the concrete UI level.

MARIA [Paterno’ et al., 2009] is a modeling language that allows abstract and concrete UI specifications. Regarding the degree of abstraction, MARIA covers different levels: the model level as well as the abstract and concrete UI level. On the abstract UI level, a data model and one or more presentations compose the UI. In turn, a data model and a dialog model, which integrate information of triggered events, constitute the presentation. Finally, the concrete UI level describes the UI in a platform-dependent manner.

IFML [Brambilla and Fraternali, 2014] is an abstract user interface modeling language that is designed to express interaction design decisions of software applications front-end. It can be used to capture the user interaction as well as UI content. In addition, it allows modeling the behavior of any UI independently of the implementation platform. In other words, IFML is an OMG standardized UI modeling language that focuses on the specification of the structure, events and the general structure of a UI. To sum up, IFML is a Platform Independent Model (PIM) level language, in Model-Driven Architecture (MDA) parlance, that is designed to provide a solution for abstract UI modeling by supporting graphical notation for the creation of visual models of the content, user interaction and controls required to specify and express the UI in an abstract manner.

Moldovan et al. [Moldovan et al., 2020] present a model-based approach for designing UIs for interactive applications. Their approach is based on an interface description language named OpenUIDL that is an open, accessible, and expressive language for designing UIs. Moreover, the proposed model-based approach covers the description of omni-channel UIs with its semantic and syntax that are specified using meta-models and JSON, respectively.

Planas et al. [Planas et al., 2021] provide a model-based approach for conversational UIs supporting the design of each interface. They make use of an existing modeling language for UI specification. In particular, the authors extend the IFML language with appropriate concepts at the meta-model level based on the extensibility rules defined in the IFML language. Additionally, their proposed approach covers the combination of the UI design with other software models that could be used within a software generation process.

3.2.1.2 Model-based and pattern-based specification

Design patterns can be part of UI specifications. In this subsection, we describe works that combine model-based approaches together with design patterns.

Molina et al. [Molina et al., 2002] present the JUST-User Interface (JUST-UI) approach that enables the use of conceptual patterns as building blocks for creating UI specifications. Within their approach, conceptual patterns represent the abstract specifications of presentation and navigation UI requirements. In JUST-UI, the UI modeling is supported by simple and complex patterns that can be observed in UI design, such as filter, navigation, and master-detail presentation patterns.

Nilsson [Nilsson, 2002] propose a UI modeling approach that is based on a combination of composite UI components and design patterns. Using this approach, the specification of UIs consists of a set of model pattern instances and mapping rules. These rules are crucial for UI modeling and represent the description of how a modeling pattern instance could be transformed to a set of interface components resulting a UI.

Radeke et al. [Radeke et al., 2006] extend the use of model-based methods with HCI design patterns. For this purpose, they develop a model-based UI development tool named Patterns In Modeling tool (PIM tool). Their tool is developed to support developers in designing UIs models through the combination of model-based and pattern-based approaches. With the PIM tool, UIs can be developed in an abstract and a conceptual way. This helps designers to handle very complex systems more easily. To build models, the PIM tool assists developers in the process of pattern application in order to make the modeling task traceable.

Engel and Märtin [Engel and Märtin, 2009] present the Pattern-Based Modeling and Generation of Interactive System (PaMGIS) framework, which is designed to support software developers by combining pattern-based and model-based approaches. This framework enables its users to design an abstract UI model used as the basis for transformation into a semi-abstract UI model. The core component of the PaMGIS framework is a pattern repository containing a collection of different patterns and pattern languages stored in XML format. To achieve UI model design and transformation processes, the PaMGIS framework suggests the use of various models including task, user, device, environment, and data architecture models.

Vanderdonckt and Simarro [Vanderdonckt and Simarro, 2010] introduce a method for UI design based on generative patterns that serves as reusable building blocks. To support their method, authors developed a software, named IDEALXML, whose purpose is to support design pattern management according to rules defined in MDA. The graphical specification of models, including task, domain and UI model, is one of the main features of IDEALXML. Based on these models, IDEALXML generates the UI specifications in UsiXML language.

Seffah et al. [Seffah, 2015a, Ahmed and Ashraf, 2007] present the Pattern-Driven and Model-Based User Interface (PD-MBUI) framework for UI development. Their framework aims at unifying the model-based approach and different patterns. It includes various components defined as follows: First, a library of patterns used as a building block for creating and transforming models. Second, a set of models including domain, user, task, environment, dialog, presentation and layout models. Finally, a Patterns Wizard tool based on XML User interface Language (XUL) and used to describe patterns, so as models. Their tool serves to guide developers in using HCI design patterns for building models. To achieve the creation of various models four principal patterns are considered, including task, feature, dialog, and presentation pattern. Each pattern represents a solution for a specific problem. For example, task patterns describe reusable task fragments included in task models. Further, in dialog modeling, patterns are used for grouping tasks to dialog view, as well as for addressing the transition between different dialog views.

Li et al. [Li et al., 2015] present a method to develop UIs on the basis of UI design patterns. Their proposed method integrates design patterns, which represent the combination of a set of structured UI elements, with the model-based approach. In their work, UI

patterns are described using an XML-Based structured pattern descriptive language named X-BSPD that depicts the UI information related to design patterns.

Märting et al. [Märting et al., 2017] propose a system, named SitAdapt, which combines model-based and pattern-based methods for the construction of interactive systems. This system extends the PaMGIS framework [Engel and Märting, 2009] in order to model the context change in the UI. The pattern repository presented in the PaMGIS framework is considered within the SitAdapt system to enable the modification of the target interface.

3.2.2 Analysis and discussion

The studied UI specification works can be divided into two different approaches. On the one hand, there exist methods that consider model-driven techniques to specify UIs. Among them, languages such as UsiXML [Limbourg et al., 2004], MARIA [Paterno' et al., 2009], TERESA [Berti et al., 2004a], and tools such as [Radeke et al., 2006] have been proposed to reduce UI development costs. Nevertheless, these traditional methods focus mainly on UI modeling at design-time, which makes the specification of UIs inflexible. Although these methods serve as a solid solution for the specification of UIs, they lack means that are required to be taken into account when it comes with the design of adaptive UIs. As a result, they do not support the adaptation of UIs since they do not explicitly support some modeling aspects, including the specification and integration of context information, as well as the UI adaptation aspects that have to be considered for the development of adaptive UIs [Brambilla and Fraternali, 2014]. Moreover, the model-based approaches are rarely used in practice since the construction, as well as the transformation of various models are a tedious and a time-consuming task that requires a multidisciplinary team with a diversity of relevant skills and knowledge about the models. In the UI modeling process, the majority of model-based approaches, including [Paterno' et al., 2009, Berti et al., 2004a] offer only a marginal support in determining mappings between different models and the model transformations are manually performed. This makes removing human interventions from the modeling process very difficult, which makes the traditional approaches non-automatic. Based on these shortcomings, it is noteworthy that the studied model-based methods, such as [Moldovan et al., 2020] and [Planas et al., 2021], do not cover the use of expert knowledge that could enable the reuse of existing knowledge about the UI design. Consequently, designers and developers are inclined to avoid these traditional methods.

On the other hand, design patterns have been assigned to various models of model-based approaches to further improve the UI quality and reduce the UI development complexity by reusing UI artifacts. According to the existing research in the specification of UIs, there are a number of works, which provide UI modeling methods that combine pattern-based and model-based approaches, such as [Nilsson, 2002], [Li et al., 2015], and [Märting et al., 2017]. Although design patterns have attracted a lot of interest in MBUID area, there still remain issues affecting UI models. For example, the conceptual patterns presented in the JUST-UI approach [Molina et al., 2002] are too abstract to promote UI knowledge reuse. In addition, the majority of works in the area of model-based UIs together with design patterns covers design-time specifications [Seffah, 2015a, Ahmed and Ashraf, 2007]. Likewise, the transformation of patterns, presented in the PaMGIS framework [Engel and Märting, 2009], is allowed at design-time and prevents pattern-base UI modification at run-time. Besides, design patterns in the PIM tool only serve as static model fragments that have to be manually manipulated by developers [Radeke et al., 2006]. This hampers the flexible use of design

pattern alternatives and restricts the mightiness of knowledge reuse. Furthermore, it is important to enhance the process of UI modeling and to improve the use of design patterns in order to automate building models as well as to foster knowledge reuse.

In spite of the plethora of UI specification approaches, works in this area have experienced a lack of interest from the industry. This lack can be explained by the poor reusability of UI model specifications. The majority of existing works serve as a solution only for design-time UI specifications. Further investigations need to be carried out in order to provide adequate tool support for applying design patterns into the specification of UI models at run-time. Additionally, it is necessary that these UI models can be interpreted on different platforms and satisfy various contexts of use. Consequentially, design pattern specification as well as run-time UI models design and adaptation need to be investigated more extensively. Finally, it is essential to improve the design patterns selection process in order to allow the specification of UIs independently of the designer's intervention. Following on from this, works related to the recommendation of design patterns are reviewed in the next section.

3.3 Design Pattern Recommender Systems

Recommender systems have become an emerging research area in different domains. In this context, several studies have presented recommender systems for retrieving design patterns. This section reviews some significant works related to the recommendation of design patterns and provides a critical analysis of the discussed works.

3.3.1 Review of design pattern recommendation

In the literature, several research studies were carried out on the recommendation of relevant design patterns for a given design problem. Each of these studies adopted different recommendation techniques.

Hamdy and Elsayed [[Hamdy and Elsayed, 2018](#)] propose an approach for automatic recommendation of software design patterns. This approach considers the text retrieval technique, where design problem scenarios are described in natural language. The recommendation approach is composed of three main steps to select suitable design patterns from a collection of 14 Gang-of-Four (GoF) patterns. It starts by preprocessing each design pattern description and design problem scenario to reduce the size and sparsity of data. Then, it relies on the indexing and feature selection step. Within this step, the collection of pattern definitions and design problem scenarios are represented using the Vector Space Model (VSM). Finally, the proposed approach applies the Cosine Similarity (CS) measures step to calculate rankings. Based on the CS measures between each design pattern vector and design problem vector, the suitable design patterns for a given design problem are selected and recommended.

Abdelhedi and Bouassidar [[Abdelhedi and Bouassidar, 2018](#)] develop an ontology-based system for recommending Service-Oriented Architecture (SOA) design patterns that are adequate for designers' modeling context. The developed system provides a questionnaire to users in order to collect their requirements. The recommender system is based on an ontology that represents knowledge about different SOA pattern problems and their corresponding solutions. This ontology is associated with SPARQL queries to search for

the appropriate SOA design pattern according to the obtained answers. After interrogating the developed ontology, SOA design patterns and their corresponding solutions that are relevant to the user problem are retrieved and recommended.

Hussain et al. [Hussain et al., 2019], present a framework to organize and select the right software design patterns for a given design problem. This framework is based on a text categorization technique to organize, first, design patterns into an appropriate number of pattern classes, then, to select the appropriate design pattern for a given design problem. To achieve these objectives, the proposed framework includes four phases: 1) the preprocessing phase applied to the design patterns' problem descriptions and the design problem. 2) The classification of design patterns based on unsupervised learning techniques. 3) The identification of design pattern class. 4) The selection of appropriate design patterns that belong to the candidate pattern class.

Celikkan and Bozoklar [Celikkan and Bozoklar, 2019] propose an approach for recommending adequate software design patterns for design problems whose description is text-based. This approach is principally based on three recommendation techniques, including text-based, case-based reasoning and question-based technique. Thus, the recommendation process, within this approach, is in three phases. It starts with the text-based recommendation technique to find matching between the description of design patterns and design problem scenarios. To achieve this purpose, CS measures between the design problem and design patterns are computed, and accordingly design patterns are ranked. Then, the case-based reasoning technique is considered to improve the obtained rankings. Finally, the question-based technique is applied where experts are asked to answer a number of questions to further enhance the rankings. Answers given by experts are used to filter and to refine the list of recommended design patterns.

Another approach for recommending design patterns is presented in [Youssef et al., 2020]. In their study, Youssef et al. propose a system that automatically recommends the relevant design pattern category. This system relies on a recommendation approach that considers the question-based technique, where software engineers are, first, asked to answer questions based on user requirements. Then, the weights of answers are measured, and accordingly the system recommends the most relevant software design pattern category.

Similarly, Naghdipour et al. [Naghdipour and Hasheminejad, 2021] propose an ontology-based approach for selecting appropriate software design patterns. This work presents a formalization of GOF design patterns based on an ontology model. The proposed approach involves two main steps. In the first step, ontology concepts are linked to WordNet and design problem scenarios are preprocessed. In the second step, the developed ontology is associated with queries to select the suitable design patterns.

3.3.2 Comparative analysis

In this subsection, we analyze the works reported in the related literature according to the following criteria:

- **Design pattern domain:** Design patterns have emerged out of different domains, such as software design patterns, SOA design patterns, and HCI design patterns.
- **Recommendation method:** Recommender systems consider various recommendation methods, namely text-based, case-based, question-based, and ontology-based

methods.

- **Automatic recommendation:** The recommendation phase may be carried out semi-automatically when the role of users is required to some extent, or fully automatically without any human expert intervention for the selection of design patterns that ought to be recommended.
- **Similarity approach:** Such recommender systems are based on the similarity of semantic or syntactic across a range of design pattern descriptions and problem scenarios.
- **Knowledge support:** Recommender systems could support the reuse of knowledge by integrating ontology models.
- **Problem input format:** Recommender systems require different problem input formats such as full-text, keywords, or questionnaires.

Table 3.1 shows a comparison of the presented works. It illustrates the main results regarding the evaluation of the studied recommendation systems considering the aforementioned criteria.

There have been many advances in the field of design patterns recommendation. Nevertheless, there are still issues to be dealt with. For instance, the recommendation domain covered in the reviewed works includes either software design patterns or SOA design patterns. Despite the growing design patterns collection in the HCI domain, existing recommendation systems do not consider this emerging domain and tend to overlook HCI design patterns. In addition, the majority of existing recommender systems rely on low-quality design problem input. Likewise, Hamdy and Elsayed propose an approach to recommend design patterns for design problems that are predefined and written briefly [Hamdy and Elsayed, 2018]. This fact may limit the set of real design problem scenarios, in a sense it restricts end users' choice regarding design problems. Moreover, many existing works adopted the text-based recommendation approach based on Natural Language Processing (NLP) methods and syntactic similarity measures [Hamdy and Elsayed, 2018, Hussain et al., 2019]. The aim of syntactic similarity measurements is to calculate the number of identical words using CS scores. Differently, the semantic similarity focuses more on the meaning and the interpretation-based similarity between design patterns and design problem scenarios. It can enable the integration of the semantic information into the recommendation process [Mu and Zeng, 2018]. Nonetheless, existing recommender systems lack the use of semantic similarity, which is essential to enhance the text-based recommendation approach and accordingly the recommendation results. Besides, some recommender systems are based on a semi-automatic recommendation strategy. For instance, the recommendation approach introduced in [Youssef et al., 2020] requires the intervention of users to answer questionnaires. Another work [Abdelhedi and Bouassidar, 2018] invites users to select the appropriate design pattern category to get the recommended SOA patterns. This strategy makes the recommendation rather semi-automatic. Finally, the use of ontology-based approaches can improve the overall quality of recommendation systems; however, few research works have taken place in the field of recommending design patterns. Existing ontology-based approaches select design patterns using queries, which is not sufficient to get the relevant ontology instances [Abdelhedi and Bouassidar, 2018, Naghdipour and Hasheminejad, 2021].

Table 3.1: Comparison of existing design pattern recommendation systems

Work	Domain	Recommendation Method	Automatic Recommendation	Semantic Similarity	Knowledge Support	Full-text	Problem Input	Format
							Question	Keyword
1 ¹	Software	Text-based	●	○	○	●	○	○
2 ²	SOA	Ontology-based	○	○	●	○	●	○
3 ³	Software	Text-based	●	○	○	●	○	○
4 ⁴	Software	Text, Case, Question-based	○	○	○	●	○	○
5 ⁵	Software	Question-based	○	○	○	○	●	○
6 ⁶	Software	Ontology-based	✖	○	●	●	○	○

Legend- ●: Supported, ○: Not Supported, ✖: Not Specified

¹[Hamdy and Elsayed, 2018]

²[Abdelhedi and Bonassidar, 2018]

³[Hussain et al., 2019]

⁴[Celikkan and Bozoklar, 2019]

⁵[Youssef et al., 2020]

⁶[Naghdiipour and Hasheminejad, 2021]

Consequently, there is a need to expand ontology-based approach with inference rules together with SPARQL queries in order to enhance the selection of relevant design pattern instances.

In general, researchers have developed different systems related to the recommendation of design patterns. However, existing recommendation systems still have some limitations and gaps. To the best of our knowledge, no attention has been devoted to providing a full automatic recommender system that automatically recommends HCI design patterns, which are essential for the design of UIs and needed for the generation of the final UIs, based on real design problem scenarios. Following on from this, works related to the generation of UIs are reviewed in the next section.

3.4 User Interface Adaptation and Generation Approaches

Many works in the area of UI adaptation and generation have been reported in the literature. This section presents a state of the art survey of this area and provides a comparative analysis of the current works.

3.4.1 Review of user interface adaptation and generation

Peissner et al. [Peissner et al., 2012] develop the MyUI framework aiming to increase accessibility through the generation of adaptive UIs. The framework performs run-time adaptations to diverse user needs and device usage by personalizing UI presentation, layout, modalities, as well as navigation path. It uses a multimodal design pattern repository that serves as the basis for creating personalized UIs. These patterns constitute the backbone of the MyUI framework and are described in a common format as introduced in [Borchers, 2008]. For creating adaptive UIs, MyUI relies on adaptation rules that are defined by design patterns and their combinations are manually handled during conception.

Gamecho et al. [Gamecho et al., 2015] present a UI generator system named Egoki. This system is designed to provide ubiquitous services for people with disabilities using a model-driven approach. Egoki incorporates three principal modules including the knowledge base, resource selector and adaptation engine. The first module considers the Egonto ontology to store, update, and maintain various models concerning user abilities, device features and adaptations. The Pellet reasoner is applied for accessing information from the knowledge base module, storing new data, as well as deducing information about adaptations. The second module is responsible for asking the ontology about users' abilities and device features in order to select the right resource for each user, its device, and service. Finally, the third module applies the needed adaptation rules to generate the final UI.

Miñón et al. [Miñón et al., 2016] present the Adaptation Integration System that focuses on integrating accessibility requirements in the process of developing Web UIs. The adaptation process relies on a repository containing rules devoted to people with special needs. These rules are incorporated into the development process across design-time and run-time. At design-time, adaptation can occur in the CRF abstraction levels. Whereas, adaptation rules at run-time are selected to create the final UI.

Hussain et al. [Hussain et al., 2018] present a model-based system that is designed for generating adaptive UIs. Their system is implemented as an Adaptive User Interface User Experience Authoring (A-UI/UX-A) tool. This tool is based on ontology models that

allow the representation of the user, context, and device. Moreover, the A-UI/UX-A tool incorporates a component for authoring adaptation rules. The construction of the A-UI/UX-A tool requires two processes: First, an offline process to create models and generate basic UI adaptation rules. Second, an online process to allow the generation of adaptive UIs.

Bouraoui and Gharbi [Bouraoui and Gharbi, 2019] introduce an approach that provides the possibility of creating UIs for multi-platforms, multi-devices, and hybrid cross platforms. It focuses on the semi-automatic generation of accessible UIs based on the MDD framework. To allow the UI generations, their approach relies on some transformations techniques. These techniques enable the transformation from abstract accessible UI models to executable UIs that meet specific accessibility requirements. Classic tools including Ecore and Eclipse Modeling Framework (EMF) are considered for metamodeling and for creating the source models and the needed transformations, respectively.

Yigitbas et al. [Yigitbas et al., 2020] provide a model-driven approach for the development of self-adaptive UIs based on the latest context of use. The authors introduce two Domain Specific Languages (DSL), including the ContextML and AdaptML for defining context properties and for modeling adaptation rules, respectively. Context information is extracted at run-time and used for selecting adaptation rules. To allow run-time UI adaptations, the proposed approach relies on several models. First, a context model is used for generating context services. Second, an adaptation model is defined for generating adaptation services. Finally, abstract UI and domain models are considered for generating the final UI.

Rieger et al. [Rieger et al., 2020] define a model-driven approach that is designed to allow the integration of accessibility concerns into the development of mobile applications. To this purpose, the MDD framework is considered along with a set of accessibility requirements for people with vision loss. Their approach aims at enhancing the quality of the generated applications, as well as reducing its development costs.

Khan and Khusro [Khan and Khusro, 2022] propose a context-aware adaptive SMS solution for drivers based on the DriverSense framework [Khan and Khusro, 2020]. Their solution is implemented as a mobile application, named ConTEXT. The purpose of their developed application is to adapt the UI according to various driving contexts. Smartphone and vehicular sensing technology are used to capture and identify various driving contexts automatically.

3.4.2 Comparative analysis

A summary of the reviewed works that provide a solution for UI adaptation and generation is outlined in Table 3.2. This Figure shows a comparison of the literature found according to the most relevant features related to the present thesis. The comparison is drawn using the following criteria:

- **Software Type:** This criterion refers to the platforms for the generated code.
- **Design environment:** This criterion describes the design environment considered in the approach, where:
 - ●: The design environment is completely fulfilled.
 - ◐: The design environment is partially fulfilled.

- ○: The design environment is not fulfilled.
- **Semantic UI modeling:** This criterion checks whether the studied works support semantic UI modeling using ontologies, where:
 - ●: The ontology is used for semantic UI modeling.
 - ○: There is no use of UI ontology models.
- **Run-time adaptation:** This criterion checks whether the adaptation is applied at design-time or run-time, where:
 - ●: The adaptation is performed at run-time.
 - ○: The adaptation is supported at design-time.
 - ⊗: The adaptation way is not specified.
- **Code generation:** This criterion shows to what extend the studied work is qualified to generate the application code, where:
 - ●: Full application code is generated.
 - ◐: Only UI code is generated or some application functions are generated.
 - ○: No code is generated.
- **Automation level:** This criterion verifies if the generation process is semi-automatic or fully automatic, where:
 - ●: The generation is fully automatic.
 - ○: The generation is semi-automatic.
- **Application example:** This criterion describes where the reviewed approach has been used, where:
 - ●: Application example is completely fulfilled.
 - ◐: Application example is partially fulfilled.
 - ○: Application example is not fulfilled.
- **Tool support:** This criterion checks whether a tool support for UI adaptation and generation is also provided by the reviewed approaches, where :
 - ●: A tool support is developed.
 - ○: There is no tool support.
- **Generation extensibility:** This criterion shows to what extend an approach is qualified to add transformations to generate code for various target languages and platforms, where:
 - ●: The approach already allows transformations to generate code for various target languages and platforms.
 - ◐: The approach allows transformations to generate code for one target language and platform, and transformations for other languages can be added.

- ○: The approach cannot add transformations to generate code for various languages and platforms.
- ⊗: The generation extensibility is not specified.

In the literature, several researchers have proposed different approaches related to UI adaptation and generation. As illustrated in Table 3.2, the reviewed approaches still have some limitations and gaps since most of them merely focus on model-based methods. Design patterns are not well handled by the available approaches towards the generation of adaptive UIs. Most of the reviewed works [Hussain et al., 2018] [Bouraoui and Gharbi, 2019] [Yigitbas et al., 2020] rely on MBUID methods. Only the MyUI system, proposed by Peissner et al., dealt with multimodal design patterns devoted to the generation of accessible UIs [Peissner et al., 2012]. Furthermore, different works including [Miñón et al., 2016], [Bouraoui and Gharbi, 2019] and [Yigitbas et al., 2020] consider UIML, Ecore or DSL modeling languages for supporting UI modeling. Nevertheless, these works do not focus on semantic UI representations using ontologies. Subsequently, existing UI adaptation and generation approaches lack the use of semantic UI modeling, which is essential to increase flexibility and to boost knowledge reuse. Thus, the UI modeling needs to be enriched with ontology-based modeling. Although run-time adaptations have gained a lot of interest, there are some issues related to adaptation rules. Likewise, authors in [Hussain et al., 2018] and [Khan and Khusro, 2022] consider only a basic level of UI adaptation rules. Similarly, rules are predefined and created at design-time within the MyUI system [Peissner et al., 2012]. Therefore, whenever a new adaptation rule needs to be included, the system has to be redeployed. In addition, using few and predefined rules can make the UI adaptation process slow and not very flexible [Miñón et al., 2016]. Regarding code generation, many works such as [Gamecho et al., 2015] and [Rieger et al., 2020] provide a full automatic generation process. Nonetheless, they are limited to the generation of parts of UI code or some application functions, thus hindering the possibility of generating full UI views. While many works address the generation of adaptive UIs, they support specific platforms: some of them such as [Peissner et al., 2012], [Gamecho et al., 2015], or [Miñón et al., 2016] generate UIs for Web applications and others including [Hussain et al., 2018], [Bouraoui and Gharbi, 2019], [Rieger et al., 2020] or [Khan and Khusro, 2022] generate UIs for Mobile applications. This limitation is due to the fact that the existing approaches are not well extensible to add various transformations in order to enable code generation for various languages and platforms. The possibility of generating adaptive UIs is studied in many works, but the questions of integrating the generated UIs into real applications and providing tool support to help developers remain open.

3.5 Concluding Remarks

In this section, we provide some concluding remarks from the related work described in the present chapter.

This chapter surveyed UI specification approaches, design pattern recommender systems, and other approaches related to the adaptation and generation of UIs. From the present chapter’s survey and discussion, three major gaps of the current state of the art have been identified, and are described below.

Table 3.2: Summary of the surveyed UI adaptation and generation approaches

Work	Software Type	Design Environment		Semantic		Run-time Adaptation	Code Gener.	Automation Level	App. Exple	Tool Support	Generation Extensibility
		MBUID	DesignPattern	UI Modeling	UI Modeling						
1 ¹	Web	○	●	○	○	●	●	●	●	●	⊗
2 ²	Web	●	○	○	○	⊗	●	●	●	○	○
3 ³	Web	●	○	○	○	●	●	●	●	○	●
4 ⁴	Mobile	●	○	○	○	●	●	●	●	●	○
5 ⁵	Mobile	●	○	○	○	○	●	●	○	○	●
6 ⁶	Web	●	○	○	○	●	●	●	●	●	⊗
7 ⁷	Mobile	●	○	○	○	●	●	●	●	●	○
8 ⁸	Mobile	●	○	○	○	●	●	●	●	○	○

Legend- ●: Completely fulfills, ○: Partially fulfills, ⊗: Does not fulfill, ○: Not Specified

¹[Peissner et al., 2012]

²[Gamecho et al., 2015]

³[Miñón et al., 2016]

⁴[Hussain et al., 2018]

⁵[Bourraoui and Gharbi, 2019]

⁶[Yigitbas et al., 2020]

⁷[Rieger et al., 2020]

⁸[Khan and Khusro, 2022]

- First, one major shortcoming of UI specification approaches is the lack of formalization and lack of knowledge reuse. Most design pattern solutions are based on informal representations. This hinders the automatic integration of design patterns in the UI modeling process.
- Second, although many systems for recommending design patterns do exist, there are only a few systems that focus on ontology-based recommendations. In addition, existing systems do not support developers throughout the whole design pattern application process, including the instantiation and the integration of HCI design patterns in the development of UIs. This limits the use of design patterns during the generation of UIs.
- Finally, the generation of adaptive UIs is studied in various works. Nevertheless, the automatic integration of the generated UIs into Web or mobile applications at run-time remains open.

To sum up, the literature review discussed in this chapter reveals that the development of adaptive UIs for mobile or Web applications is an interesting research topic, which has received considerable attention over recent years. Despite the numerous literature found, there are still significant challenges that need to be addressed, such as appropriate application of expert knowledge, or appropriate mechanisms to guarantee the use of design patterns for adaptive application development. The research presented in this thesis is a step forward to tackle these challenges. It focuses on the combination of model-based development with design patterns to enable the use of expert knowledge for adaptive application development to make an original contribution to the HCI field. The conclusions reached throughout this chapter emphasize the need for the development of a new framework to support the use of design patterns for the design and generation of adaptive UIs as well as to address the deficiencies of the existing works. In the next chapter, the framework defended in this PhD thesis is presented.

AUIDP: A Framework for the Design and Generation of User Interfaces

4.1 Introduction

The present thesis faces the development of UIs for mobile and Web applications by considering their design, development and run-time adaptations. In the context of this PhD thesis, we propose a framework that supports the design and generation of UIs using expert knowledge. The aim of this framework is to enable the use of HCI design patterns to capture expert knowledge in ways that it can be reused to solve design problems. Within the proposed framework, models are used both to centralize knowledge of design patterns and to drive appropriate design solutions. These design solutions are considered to create and adapt UIs in different context situations. In this sense, the design and generation of UIs is performed automatically by the proposed framework with the goal of achieving usable interfaces for mobile and Web applications.

The remainder of this chapter is organized as follows: First, Section 4.2 provides an overview of the main building blocks that constitute the proposed framework. Then, Section 4.3 describes the framework from an architectural point of view. Next, Section 4.4 relates implementation details to make automatic and run-time UI development. Finally, Section 4.5 concludes this chapter with a discussion and brief summary.

4.2 Main Building Blocks

The framework proposed in this thesis is named Adaptive User Interface Design Pattern (AUIDP) [Braham et al., 2020, Braham et al., 2021a]. It seeks to design and generate UIs for mobile and Web applications using expert knowledge. One of the central features of the present framework is the ability to develop applications using HCI design patterns that incarnate expert knowledge. It offers new capabilities for selecting HCI design patterns and their use for the development of adaptive UIs. Therefore, the AUIDP framework brings

a solution for representing knowledge related to HCI design patterns, for selecting the relevant design patterns, and for generating adaptive UIs automatically.

Figure 4.1 presents the overall view of the AUIDP framework decomposed in its main building blocks. From a methodological perspective, the building blocks, shown in Figure 4.1, can be split into two major levels: design-time and run-time. The first level serves to create models that represent knowledge about HCI design patterns. Once these models are built, they are used within the second level of the AUIDP framework in order to achieve the development of different UIs. The second level aims to use the created models to allow the selection of HCI design patterns and to enable the generation of UIs from the design solutions provided by the selected patterns. At this level, the AUIDP framework can be considered as the assembly of two systems, including the user Interface Design Pattern Recommender (IDEPAR) system [Braham et al., 2021b] and the user Interface Code Generator using DDesign Patterns (ICGDEP) system [Braham et al., 2021c]. The IDEPAR system focuses on the recommendation of design patterns, whereas the ICGDEP system supports the generation of the UI source code.

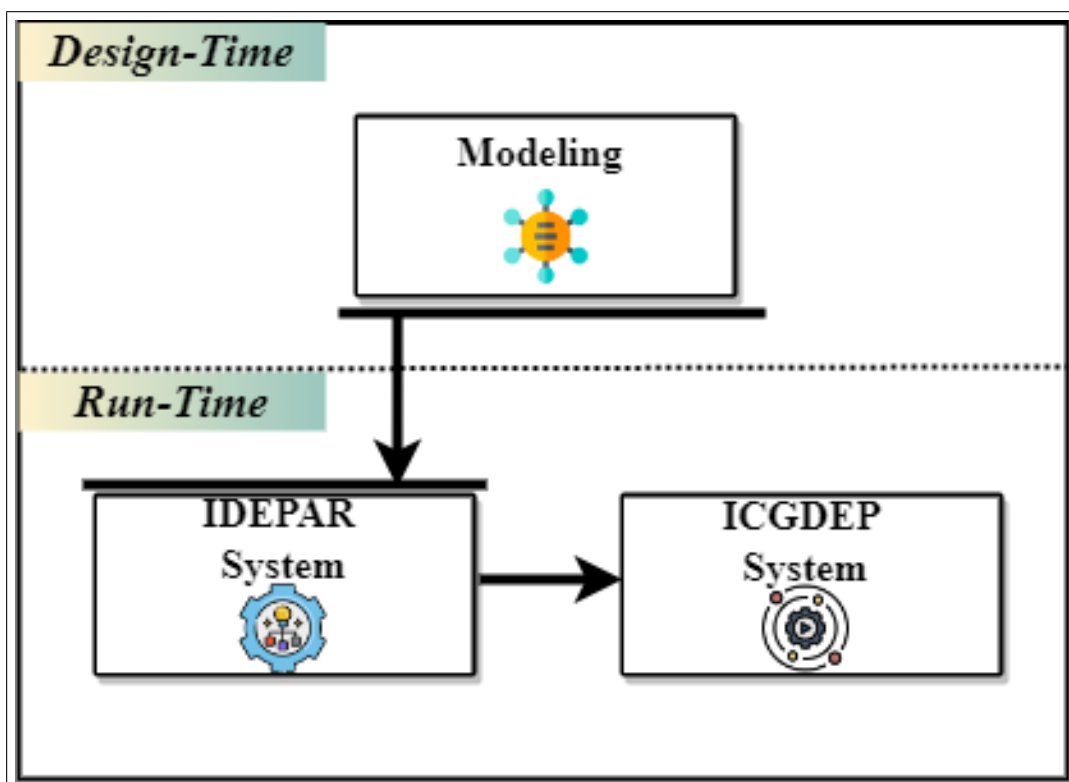


Figure 4.1: Main building blocks of the AUIDP Framework

A brief description of each building block that constitutes the AUIDP framework is given below.

- **Modeling:** In this stage, the different models that represent knowledge useful for the design of UIs are created. Specifically, models of design patterns along with UI and user profile are defined following a specific development methodology. In these models, relations between concepts are also included to aid in the generation of

feasible interface design. Once these models are built, they should be integrated in the modeling environment of the AUIDP framework. Figure 4.2 depicts the process to be followed to create models of the present modeling stage. More details regarding the specification methodology to represent these models are outlined in Chapter 5.

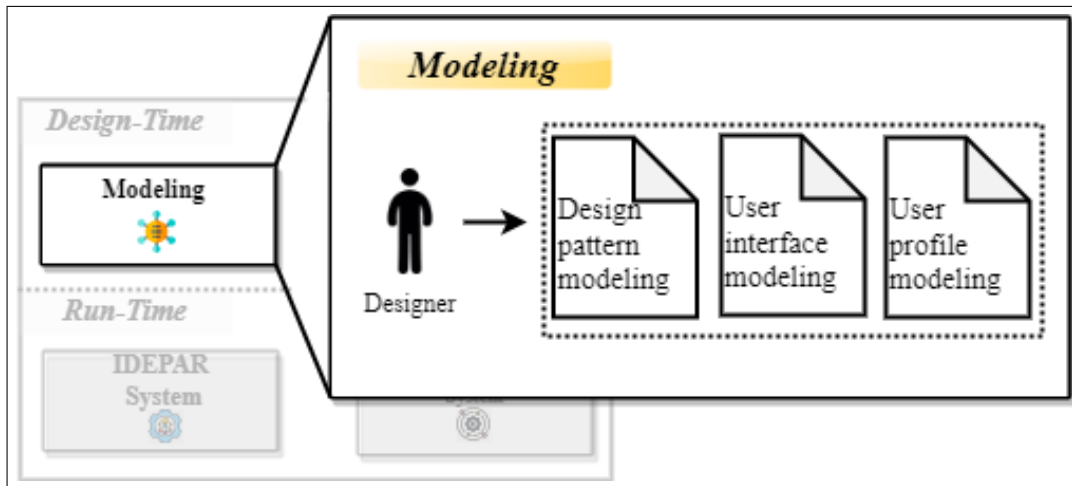


Figure 4.2: Overview of the modeling stage

- **IDEPAR system:** This first system is based on the models defined at design-time. It is in charge of recommending automatically at run-time the most relevant HCI design patterns for a given design problem. To this end, the IDEPAR system follows a recommendation approach for selecting appropriate design patterns needed for the second system. Further details about the IDEPAR system are provided in Chapter 6.
- **ICGDEP system:** This second system is based on the HCI design patterns recommended by the IDEPAR system to build the UI. It mainly aims at generating automatically the UI source code from the design solutions provided by the recommended patterns. To achieve this, the ICGDEP system relies on a generation method that enables source code generation for mobile and Web applications. This system is detailed in Chapter 7.

4.3 Framework Architecture

In this thesis, we aim to provide an environment for multidisciplinary teams to design and generate UIs in a consistent way. To this purpose, we have identified the following main aspects that characterize the proposed framework:

- **Open and accessible:** The framework puts design patterns at the fingertips of designers or software developers so they could be used for interface development.
- **Modular:** The framework is based on models that integrate different HCI design pattern instances to ensure modularity.
- **Code reuse:** The framework offers mechanisms to automate the UI development process. This fact fosters code reuse and thus reduces the code that has to be developed.

The AUIDP framework aims to support the design and generation of UIs using a hybrid approach that combines model-based UI development with the pattern-based method. Such a framework relies on the idea that, to model the UI of the target application, one can use the combination of model fragments. These fragments integrate implicit knowledge about HCI design patterns that constitute the core of the present framework. The overall architecture of the AUIDP framework for UI design and generation is depicted in Figure 4.3. The present framework is based on two systems, namely IDEPAR and ICGDEP. These systems interact among them to solve the given design problems by generating adaptive UIs.

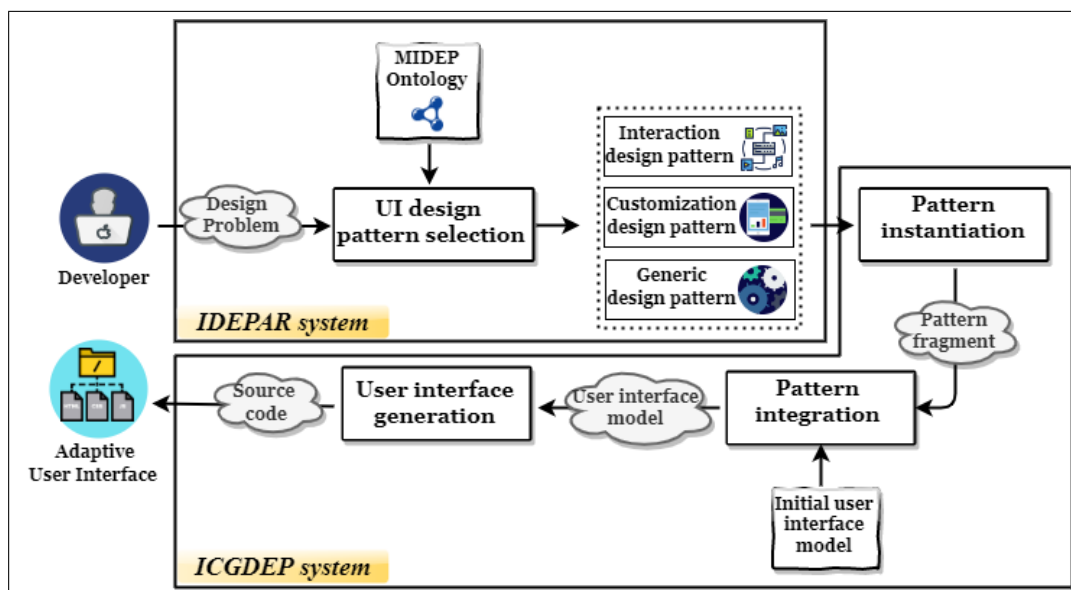


Figure 4.3: Architecture of the AUIDP framework

First, the IDEPAR system requires as input the description of design problem scenarios to select the most relevant HCI design patterns. Figure 4.4 outlines the main components incorporated in the IDEPAR system. To achieve the selection of relevant patterns, the IDEPAR system entails strategies that allow querying the created models and retrieving the appropriate design patterns. As a result, the IDEPAR system provides a list of the recommended HCI design patterns, which are classified into three principal categories, including interaction, customization, and generic categories.

- The interaction category: Refers to design patterns that offer design solutions related to UI elements.
- The customization category: Includes design patterns that provide design solutions about the look and feel of the UI.
- The generic category: Defines the miscellaneous category that refers to other design patterns, which could not be presented in the first and the second category.

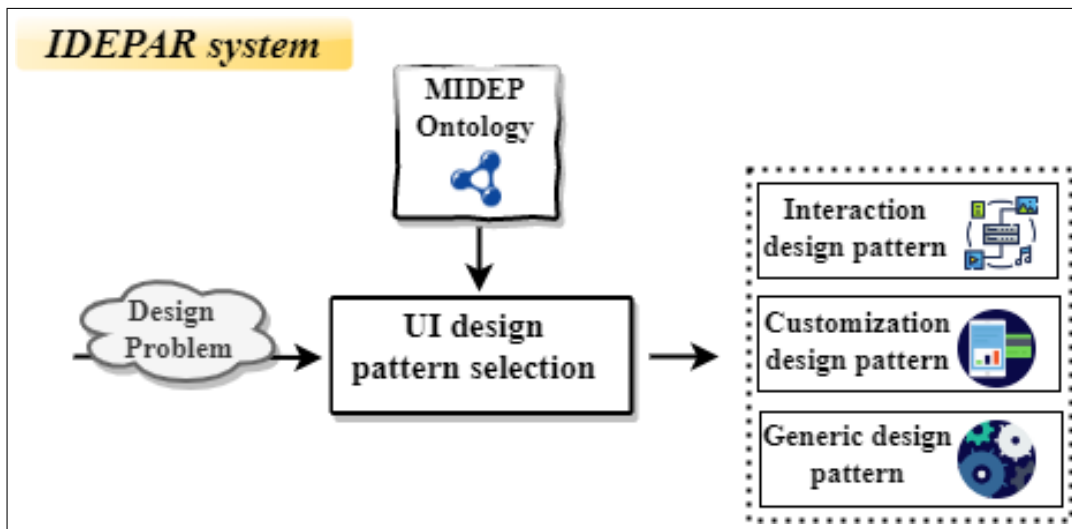


Figure 4.4: Components of the IDEPAR system

Second, the ICGDEP system focuses on generating the UI source code using the HCI design pattern selected by the IDEPAR system. As illustrated in Figure 4.5, the ICGDEP system is composed by three main components, including pattern instantiation, pattern integration, and source code generation. A brief description of each component is provided below:

- Pattern instantiation: This component allows recasting the general form of each selected HCI design pattern into a concrete form.
- Pattern integration: This component aims at integrating pattern fragments into a UI model.
- Source code generation: This component focuses on generating UI source code for a target application automatically.

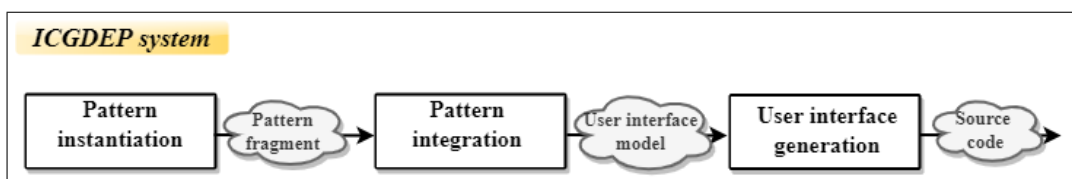


Figure 4.5: Components of the ICGDEP system

4.4 Framework Implementation

The present AUIDP framework aims to automatically design and generate UIs using HCI design patterns in such a way that the proposed framework allows the adaptation of different UIs for mobile and Web applications at run-time to satisfy user's profile and requirements. In the following subsections, we first provide an overview of the framework implementation. Then, we describe the technical architecture of the proposed AUIDP framework.

4.4.1 Implementation overview

The implementation of the global AUIDP framework is performed in order to allow the automatic generation of adaptive UIs. The generated interfaces are adapted at run-time and fit with real user needs. The implementation of the proposed AUIDP framework is achieved by building the framework's main systems, including the IDEPAR and the ICGDEP system. These systems provide various functionalities that are required to support the framework capabilities. Specifically, the resulting functionalities are related to the recommendation of HCI design patterns and the generation of interface code granted by the IDEPAR and the ICGDEP system, respectively. Figure 4.6 shows the process to be followed to implement the AUIDP framework.

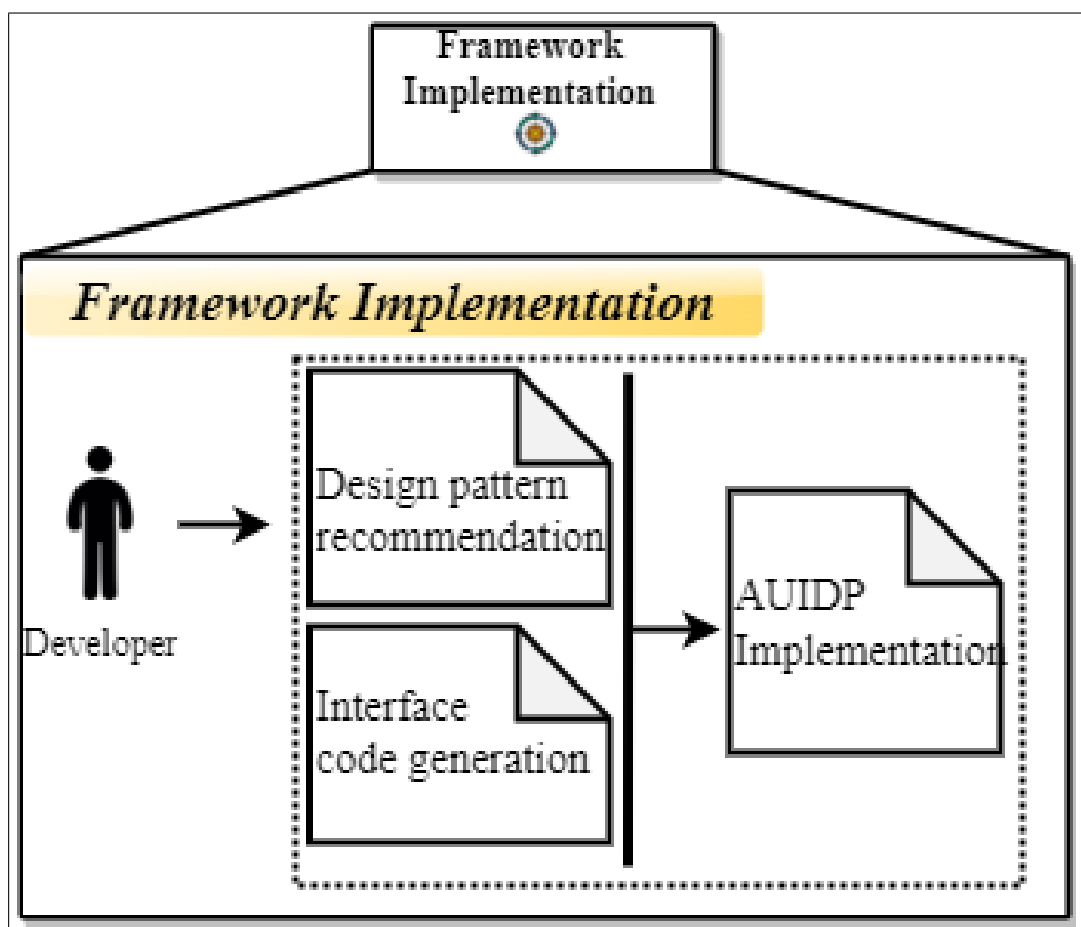


Figure 4.6: Overview of the framework implementation

4.4.2 Technical architecture of the AUIDP framework

In this thesis, the present AUIDP framework is built over infrastructures that are prepared for automatic UI adaptations for a target application at run-time. To this end, we propose two run-time architectures. The first architecture concerns the development of mobile applications, whereas the second architecture targets the development of Web applications. In the following subsections, we describe the execution environments that we have implemented for the aforementioned run-time architectures.

4.4.2.1 Execution environment for mobile applications

To illustrate the execution environment for mobile applications, we choose the Ionic framework. In this sense, the present environment exploits various Ionic concepts for characterizing the final UI that is designed and generated by the AUIDP framework.

As shown in Figure 4.7, apart from the AUIDP systems, including the IDEPAR and the ICGDEP system, the execution environment for mobile applications requires a module for generating modular plugins. Each generated plugin is packaged as an Ionic module and all exported plugins are installed in the backend of the Ionic application. In this case, the present AUIDP framework supports the development of Ionic applications through reusable and flexible plugins. The steps considered to allow the adaptation of the final interface for mobile applications are summarized below:

- **Step 1:** The context manager module, included in the frontend of the Ionic application, manages the design problem information of the current user and sends these information to the plugin selector module.
- **Step 2:** According to the detected design problem, the plugin selector module selects the appropriate plugin that handles a separate UI component for specific design problems.
- **Step 3:** Once appropriate plugins are selected, the plugin activator module activate them. Consequently, the UI components, which belong to the activated plugins, are dynamically displayed to the end user.

To sum up, this first execution environment highlights the capability of the AUIDP framework to allow the adaptation of mobile UIs at run-time. The process of integrating the UI source code, generated by the AUIDP framework for the current context of use, is achieved by extending Ionic applications through plugins. These plugins implement the interface generated by the proposed framework.

4.4.2.2 Execution environment for Web applications

The second execution environment, illustrated in Figure 4.8, is implemented based on the Angular framework. This environment uses Angular framework concepts to characterize Web UIs that are designed and generated by the AUIDP framework. Apart from the AUIDP systems, the present execution environment needs an additional module for bundling the generated UI source code. The source code provided by the framework sub-systems consists of HTML, SCSS and TypeScript files.

As shown in Figure 4.8, the AUIDP framework bundles these files in one single JavaScript file using the bundler Webpack module. At run-time, the interface is adapted according the current user needs. The adaptation of the final interface for Web applications consists of the following main steps:

- **Step 1:** The context manager module, presented in the frontend of the Angular application, manages the design problem information of the current user and sends these information to the adaptation launcher module.

- **Step 2:** The adaptation launcher module sends the detected design problem to the AUIDP framework. First, the IDEPAR system selects the most relevant HCI design patterns to solve the detected design problem. Then, the ICGDEP system generates the source files that are used in the following step.
- **Step 3:** The bundler Webpack module bundles the resulting files in one single JavaScript file that is stored in a repository of bundles.
- **Step 4:** Once the bundle file is created and stored, the component creator module, presented in the backend of the Angular application, achieves the following sub-steps: (i) First, it imports the bundle file from the repository. (ii) Second, it compiles the imported bundle. (iii) Finally, it injects the created component that corresponds to the compiled bundle in the interface display module.
- **Step 5:** The interface display module adapts the current interface by displaying to the end user the UI components that belong to the injected bundle.

To sum up, this second execution environment shows the capability of the AUIDP framework to allow the adaptation of Web UIs at run-time. The process of integrating the UI source code, generated by the AUIDP framework for the current context of use, is achieved by extending Angular applications through bundles.

4.5 Concluding Remarks

In this chapter, we have introduced the AUIDP framework from two related points of view: First, we have presented the high-level view of the building blocks involved in the AUIDP framework. Second, we have presented the proposed framework from an architectural point of view. Additionally, we have described the AUIDP framework's implementation, which is achieved using two infrastructures that ensure the automatic UI adaptations for mobile and Web applications at run-time.

In the following three chapters, we provide a detailed description of the building blocks identified in the AUIDP framework. First, Chapter 5 introduces the specification method for creating models involved in our framework. Then, Chapter 6 describes the IDEPAR system, which allows the automatic recommendation of the most relevant HCI design patterns. Finally, Chapter 7 presents the ICGDEP system, which enables the automatic generation of the UI source code for a target application.

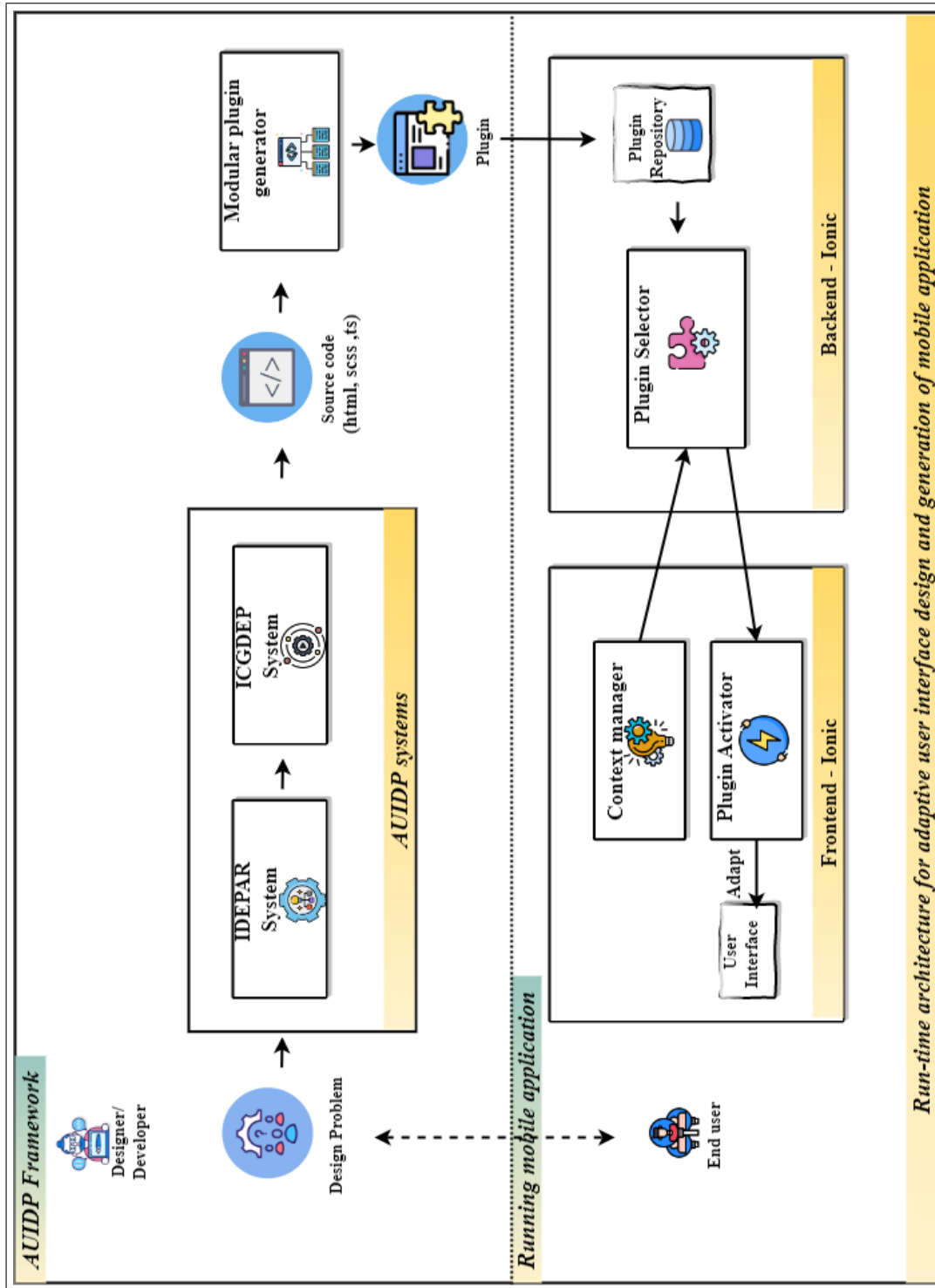


Figure 4.7: Execution environment for mobile applications

4. AUIDP: A FRAMEWORK FOR THE DESIGN AND GENERATION OF USER INTERFACES

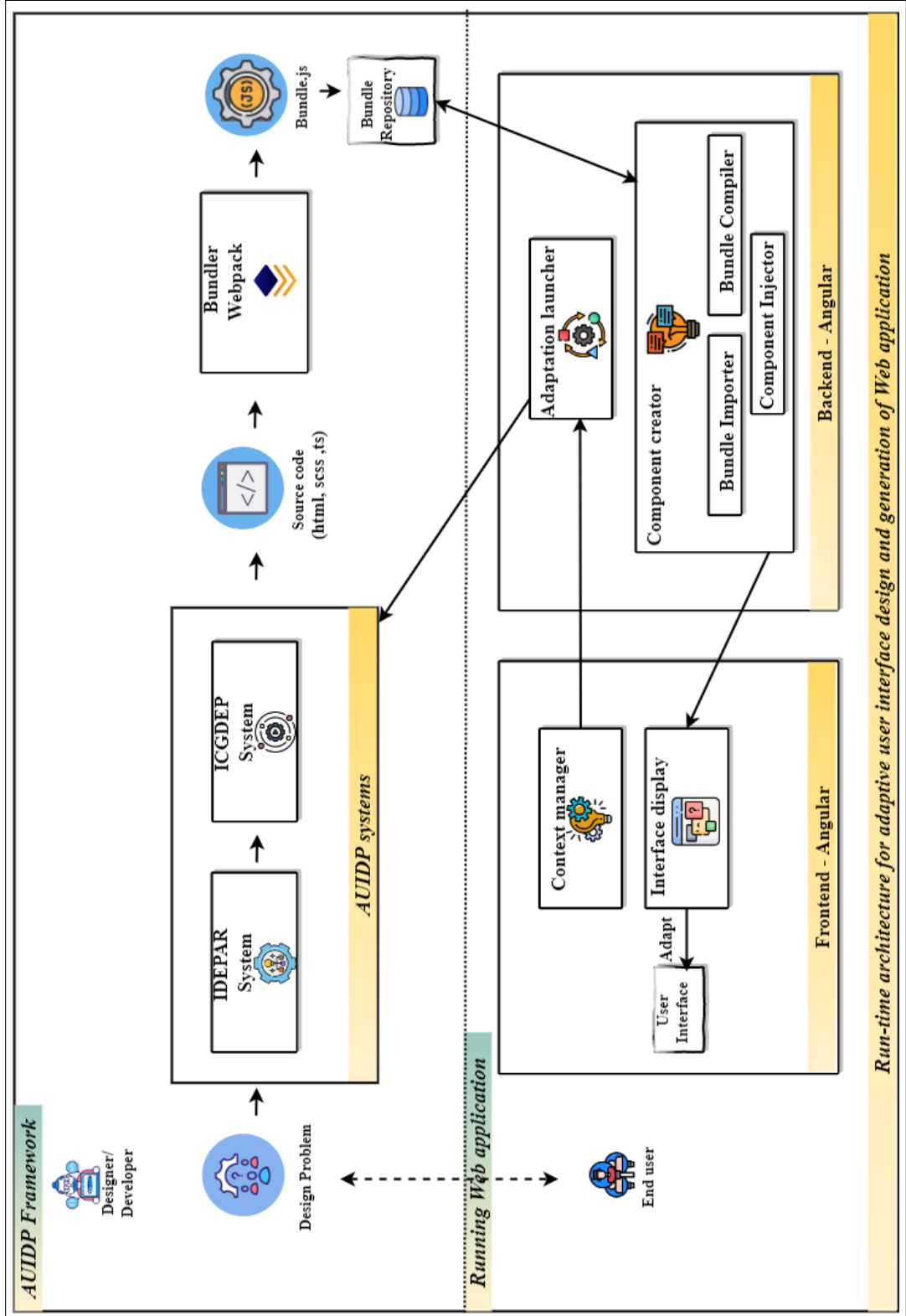


Figure 4.8: Execution environment for Web applications

Design Pattern Specification Method

5.1 Introduction

The present thesis focuses on the use of design patterns for the design and generation of UIs. In addition to the textual representation, a formal representation of HCI design patterns is proposed in this dissertation in order to facilitate the management and the use of various design patterns, as well as to offer an infrastructure for applications bearing pattern-based design processes. To this end, ontology is used to provide a semantic description of HCI design patterns that provide relevant design solutions. Building an ontology, which represents knowledge about design patterns, requires a methodological approach.

Throughout this chapter, we describe the design pattern specification method that is applied to build the proposed ontology model. The present chapter starts by detailing in section 5.2 the overview of the specification method. Afterwards, the MIDEP ontology development phases are described in Section 5.3. Finally, Section 5.4 gives some concluding remarks.

5.2 Design Pattern Specification Overview

To establish the process that turns traditional text-based presentation into formal HCI design pattern representation, we relied on the use of ontologies. As introduced in Chapter 2, ontologies are promising to undertake a formal representation [Henninger and Corrêa, 2007], as well as to apply reasoning techniques over this kind of representation. In this thesis, HCI design patterns are formally represented using ontologies. In this sense, a Modular user Interface Design Pattern (MIDEP) ontology [Braham et al., 2020, Braham et al., 2021a] is proposed to afford an open and extensible HCI design pattern specification. MIDEP is an ontology that includes all necessary specifications to provide a standard form of HCI design patterns. The present ontology is specifically conceived to cover concepts related to HCI design patterns. In order to ensure modularity and cover knowledge regarding user requirements, the MIDEP ontology is also extended with concepts related to UIs and the

end user. These concepts are included to allow the representation of knowledge regarding UI elements and user profile.

The transition from informal representation to a formal one is achieved by applying an ontology engineering methodology. From the methodologies presented in Chapter 2, there exist several methods such as METHONTOLOGY [López et al., 1999], DILIGENT [Pinto et al., 2004], and On-To-Knowledge [Sure et al., 2004] for ontology development. These methodologies propose pertinent guidelines for designing and building ontologies; however, they all have some limitations for reusing and re-engineering existing ontologies [Gómez-Pérez, 2004]. To accomplish these limitations, the NeOn methodology [Suárez-Figueroa et al., 2012] is selected and considered as the most appropriate one to cover the need of the MIDEP ontology development process. In this thesis, the NeOn methodology is adopted based on the following considerations. First, it is a methodology that serves to re-engineer non-ontological resources into ontologies and to reuse existing ontologies. This enables the development of modular ontologies. Second, it is a scenario-based methodology that encompasses a set of guidelines for different activities and processes, which emphasizes its flexibility and adaptability to various development scenarios.

In line with the spirit of the NeOn methodology, three scenarios and some specific activities have been combined to contribute to the development of the MIDEP ontology. As shown in Figure 5.1, scenario 1 (From specification to implementation), scenario 2 (Reusing and re-engineering non-ontological resources), and scenario 4 (Reusing and re-engineering ontological resources) are considered for building the MIDEP ontology. A detailed description of each scenario is presented below.

- **Scenario 1:** From specification to implementation: is the basic scenario that is involved in any ontology development since it can be combined with the rest of scenarios. It mainly consists of the development of the ontology network from scratch considering the following core activities:
 - **Specification:** This activity refers to the definition of a set of requirements that the ontology has to fulfill. These requirements are gathered in a document named ORSD. In particular, this document includes information about the ontology purpose and scope, its intended end users and its implementation language. This specification document also requires the identification of the non-functional and functional requirements, which the ontology should satisfy, using natural language in the form of competency questions, and the definition of a pre-glossary of terms extracted from competency questions. In particular the ORSD allows to:
 - * Identify and reuse the knowledge resource to be represented in the target ontology.
 - * Verify the resulting ontology regarding the ontology requirements.
 - **Scheduling:** Once the ontology requirements have been identified, the rest of scenarios and activities needed for the ontology development have to be scheduled. This activity includes finding out the ontology life cycle model, selecting the adequate processes and activities, prioritizing the selected processes and activities, and establishing the correspondence between processes and activities, and the ontology life cycle model.

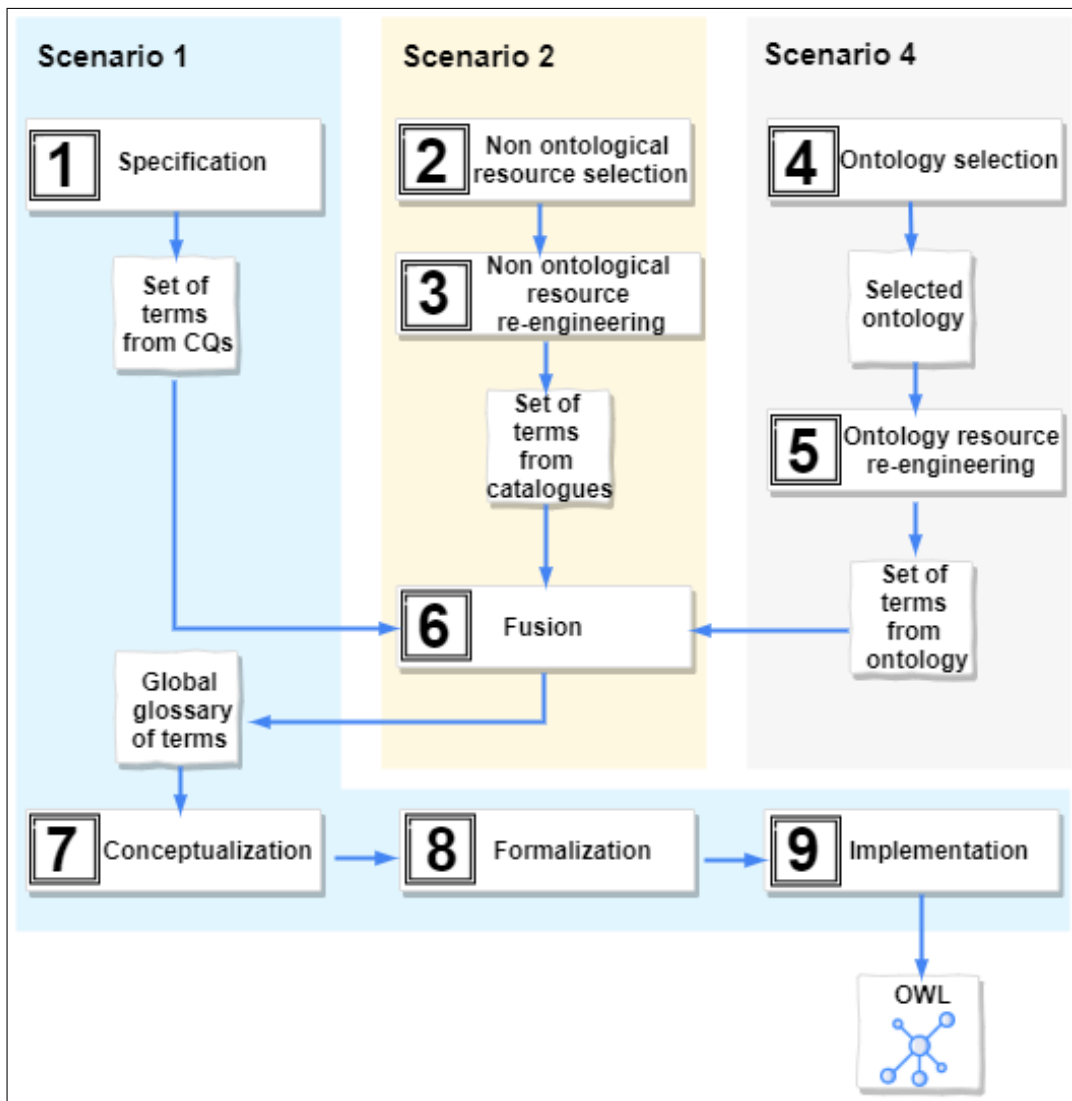


Figure 5.1: Scenarios for building the MIDEP ontology

- **Conceptualization** It is the process of creating an abstract view of a specific domain represented using a set of interconnected concepts. It refers to the activity of organizing and structuring knowledge including concepts and relationships into a conceptual model that identifies the problem and its solution considering the domain vocabulary defined in the specification activity.
- **Formalization:** This activity refers to the transformation of the conceptual model, obtained in the conceptualization activity, into a semi-computable model.
- **Implementation:** Once the semi-computable model is defined, the purpose of this activity is to implement the obtained model in a formal ontology language resulting a computable model.
- **Scenario 2:** Reusing and re-engineering non-ontological resources: this scenarios deals with reusing and re-engineering existing Non-Ontological Resources (NORs) related to the knowledge domain of the target ontology. These non-ontological

resources are considered as knowledge resources whose semantic has not been previously formalized as an ontology, such as glossary, lexicon, dictionary, taxonomy and thesaurus. The non-ontological resource reuse is the first process toward scenario 2 and it is divided into the following activities:

- **Activity 1:** Non-ontological resources search: This activity aims to search and retrieve non-ontological resources from highly reliable Web sites, domain-related sites, and resources within organizations according to the requirements of the ORSD.
- **Activity 2:** Candidate non-ontological resources assessment: This activity concerns the assessment of the non-ontological resource candidates considering a set of criteria including coverage, precision, and consensus criteria.
- **Activity 3:** The most appropriate non-ontological resources selection: The goal of this activity is to select the appropriate non-ontological resources from the set of candidate non-ontological resources resulted from the previous activity.

After reusing the non-ontological resources, ontology developers have to deal with the non-ontological resource re-engineering process, which is defined as the process of transforming the non-ontological resource into an ontology considering three main activities:

- **Activity 1:** Non-Ontological resource reverse engineering: The goal of this activity is to analyze the selected non-ontological resources, identify the corresponding components, and create the resource representation at different levels of abstraction.
 - **Activity 2:** Non-Ontological resource transformation: The second activity carried out within the re-engineering process is the transformation of non-ontological resources whose objective is to generate a conceptual model from the selected non-ontological resources.
 - **Activity 3:** Ontology forward engineering: The purpose of this activity is to integrate the non-ontological resources transformed in the previous activity to the ontology network, and thus it allows to generate a new implementation of the ontology.
- **Scenario 4:** Reusing and re-engineering ontological resources: This scenario unfolds in cases where an ontology resource needs some extension and modification to serve to the ontology requirement specification activity and meet the intended purpose. To support this scenario, ontology developers have to perform the ontological resource reuse process that consists of four main activities as follow:
 - **Activity 1:** Ontology search: The goal of this activity is to look for existing ontologies, in semantic search engines, that could meet the need of the target ontology. The output of this activity is a set of candidate ontologies.
 - **Activity 2:** Ontology assessment: The objective of the ontology assessment is to find out which ontological resources, obtained in Activity 1, meet the

requirements identified in the ORSD and to decide their usefulness for the ontology development.

- **Activity 3: Ontology comparison:** The goal of this activity is to compare the ontological resources considering a set of criteria such as reuse economic cost, code clarity, and content quality.
- **Activity 4: Ontology selection:** The objective of the ontology selection is to find out the most appropriate ontology, from the set of candidate ontology resources obtained in Activity 3, for the development of the target ontology network.

The second process defined in this scenario is about reengineering ontology resources and it consists of the following three activities that are applied to the selected ontology:

- **Activity 1: Ontological resource reverse engineering:** The goal of this activity is to build an initial ontology conceptual model from its source code.
- **Activity 2: Ontological resource restructuring:** The objective of this activity is to correct and organize knowledge that are defined in the conceptual model obtained from the previous activity, and to detect the missing knowledge. The output of this activity is a new conceptual model.
- **Activity 3: Ontological resource forward engineering:** The purpose of this activity is to generate a new implementation of the selected ontology based on the conceptual model obtained in Activity 2.

5.3 Ontology Development Phases

5.3.1 Specification

The output of this activity is the MIDEF ontology [ORSF](#) where information about the purpose, scope, implementation language, intended uses, requirement, and pre-glossary of terms of the target ontology is described. [Table 5.1](#) and [Table 5.2](#) present an example of the MIDEF ontology requirement specification document derived from the specification activity.

5.3.2 Scheduling

The development process of the MIDEF ontology fits with the six-phase waterfall ontology network life cycle model [[Suárez-Figueroa et al., 2012](#)] since it extends the 4-phase cycles (initiation, design, implementation, and maintenance phase) with a reuse phase and a re-engineer phase. [Figure 5.2](#) presents the selected ontology life cycle along with the aforementioned NeOn scenarios (scenario 1, scenario 2, and scenario 4), their corresponding activities, and the MIDEF ontology modules.

5.3.3 Knowledge resource reuse and re-engineering

As shown in [Figure 5.2](#), the second and third phases for the development of the MIDEF ontology concerns the reuse and re-engineering of knowledge resources. These phases are summarized in the following subsections.

Table 5.1: Excerpt of the MIDEP Ontology Requirement Specification Document: Part I

<p>Purpose</p> <ul style="list-style-type: none"> - A modeling solution to tackle recurring design problems related to user interfaces. - An adequate framework for the design and generation of user interfaces. - Services for recommending design patterns in a sustainable matter. - A modular conception of design patterns. <p>Scope</p> <p>The MIDEP ontology has to be created based on the HCI domain.</p> <p>Implementation</p> <p>The ontology is Formalized in the OWL DL Language and implemented using the Protégé modeling tool.</p> <p>Indented End-User</p> <p>User1. Interface designer, Software developer.</p> <p>User2. All users of Web and mobile applications.</p> <p>Indented Uses</p> <p>Use1. To help the process of designing and generating user interfaces.</p> <p>Use2. To help the process of identifying design problems.</p> <p>Use3. To identify the relevant design patterns for a specific design problem.</p> <p>Use4. To support search of information about the design patterns that compose a user interface.</p> <p>Use5. To help the process of recommending design patterns.</p> <p>Ontology requirements</p> <p>a) Non-Functional Requirements</p> <p>NFR1. The ontology should be described and documented in English.</p> <p>NFR2. The ontology should follow a modular architecture.</p>

Table 5.2: Excerpt of the MIDEP Ontology Requirement Specification Document: Part II

<p>b) Functional Requirements: Groups of Competency Questions</p> <p>CQG1. Design Pattern (7 CQs)</p> <p>CQ1. What are the design patterns contained in the ontology?</p> <p>CQ2. What are the concepts modeled in the ontology that can be used to categorize design patterns?</p> <p>CQ3. What are the concepts represented in the ontology that model a design problem?</p> <p>CQ4. What are the design patterns that provide a solution to the problem of colorblindness?</p> <p>CQ5. What are the design patterns that provide a solution to the problem of low vision?</p> <p>CQ6. What are the elements that compose a design pattern?</p> <p>CQ7. What are the solutions that a design pattern can provide?</p> <p>CQG2. User Interface (3 CQs)</p> <p>CQ1. What are the elements that compose a user interface?</p> <p>CQ2. How is a user interface identified?</p> <p>CQ3. What are the characteristics of a user interface?</p> <p>CQG3. User Profile (4 CQs)</p> <p>CQ1. What is the profile of a user?</p> <p>CQ2. What impairments or combination of impairments does a user have?</p> <p>CQ3. What preferences does a user have?</p> <p>CQ4. What activities can a user perform?</p> <p>Pre-Glossary of terms</p> <p>a) Terms from Competency questions + Frequency</p> <p>Design pattern (Freq. = 6), solution (Freq. = 2), design problem (Freq. = 3), user interface (Freq. = 3), user (Freq. = 3), impairment (Freq. = 1), preference (Freq. = 1).</p> <p>b) Terms from Answers + Frequency</p> <p>Design pattern name (Freq. = 1), condition (Freq. = 1), user interface component (Freq. = 1), visual impairment (Freq. = 1), colorblindness (Freq. = 1).</p>
--

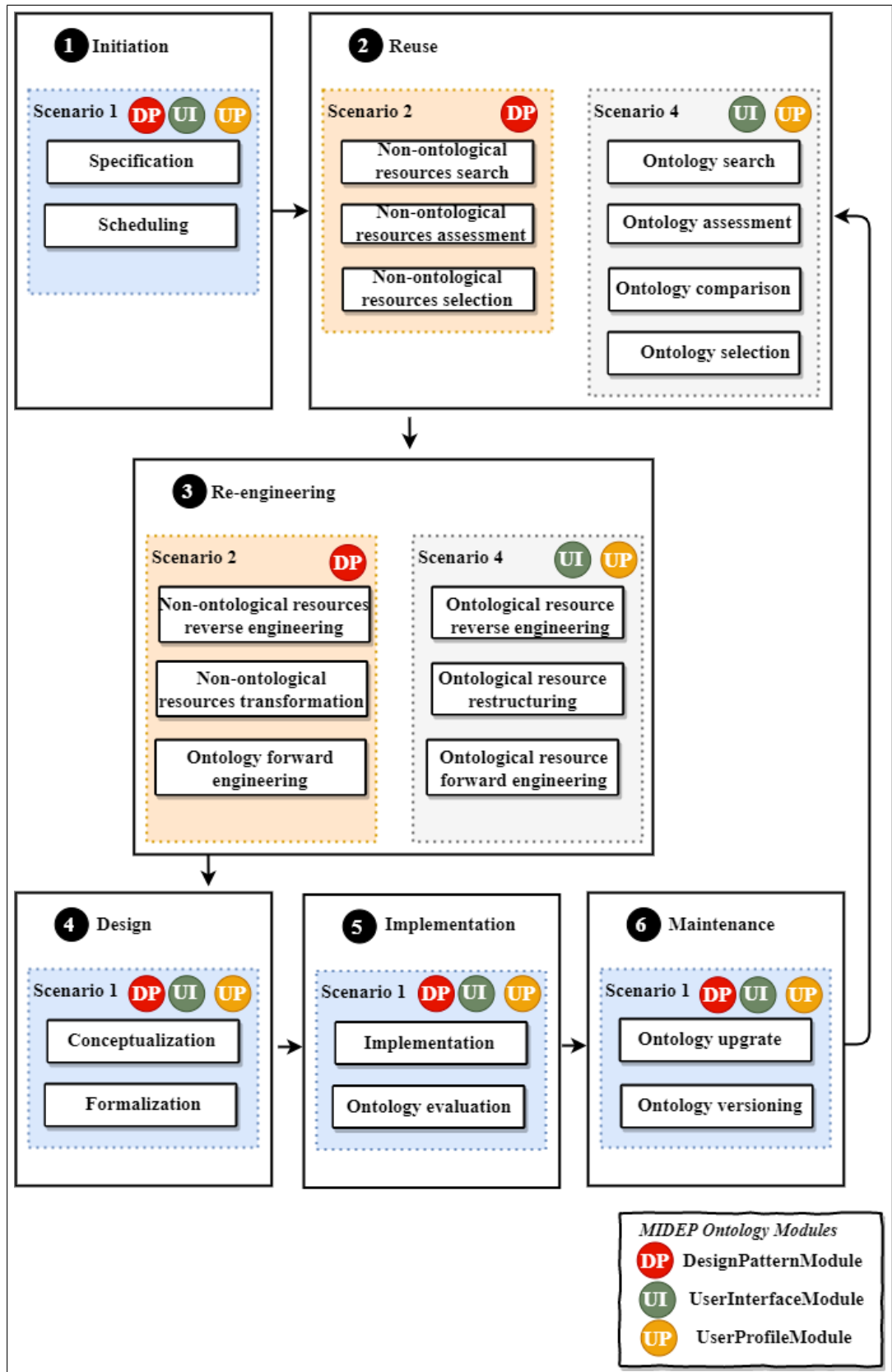


Figure 5.2: MIDE ontology life cycle

5.3.3.1 Knowledge resource reuse

The second phase within the MIDEF ontology life cycle concerns the reuse of existing knowledge resources including non-ontological and ontological resources.

- Non-ontological resource reuse

The non-ontological resource reuse is the first process toward scenario 2. This process emphasizes the reuse of non-ontological resources that cover the desired terminology using terms that have the highest frequency in the ORSD (Design pattern, Freq. = 6). To fulfill this process, we tried first to find out non-ontological resources related to the desired terminology following two main stages:

- Stage 1: Highly reliable Website search: In this stage, we search reliable Web sites that publish design pattern collections and catalogues.
- Stage2: Scientific literature search: In this stage, we conducted a literature review about design pattern collections and catalogues developed in scientific documents.

Then, we tried to perform the assessment of non-ontological resources candidates and select the most appropriate one resulting in a set of non-ontological resources found in reliable Websites and in the scientific literature. The selected Websites and existing works that provide relevant HCI design patterns are illustrated in Table 5.3 and Table 5.4, respectively.

Table 5.3: Results of the NOR from Websites

Work	Identified Design Patterns
1 ¹	This Web site presents a pattern library for interaction design.
2 ²	This Web site is about a library of UI design patterns.
3 ³	This Web site provides a list of patterns for effective interaction design, which are extracted from Tidwell's library [Tidwell, 2010]
4 ⁴	This Web site includes patterns that are introduced in [Peissner et al., 2012].

^a<http://www.welie.com/patterns/index.php>

^b<https://ui-patterns.com/>

^c<https://www.oreilly.com/>

^d<http://myuipatterns.clevercherry.com/>

Table 5.4: Results of the NOR from the scientific literature

Website	Description
[Tidwell, 2010]	125
[Neil, 2014]	70
[Wetchakorn and Prompoon, 2015]	15
[Nilsson, 2009]	10
[Kultsova et al., 2016]	3

- **Ontological resource reuse**

The ontological resource reuse is the first process in scenario 4 of the proposed ontology development method. The goal of this process is to reuse existing ontological resources considering the ontology requirements presented in the ORSD and the documentations of ontologies to be reused. In this process, various ontological resources were searched and assessed taking into account the ontology content and granularity. The comparison between the candidate ontology resources is based on two main criteria, including the reuse cost and the reliability level of the candidate ontologies. As a result of this process, the following ontologies are selected:

- The SOUPA ontology [Chen et al., 2005]: This ontology focuses on representing the context in pervasive environments.
- The ACCESSIBILITIC ontology [Mariño et al., 2018]: This ontology aims at modeling knowledge about the user's disability.
- The GUMO ontology [Heckmann et al., 2005]: This ontology represents knowledge about the user's profile.
- The OAFE ontology [Dandan et al., 2018]: This ontology focuses on describing knowledge regarding user's activities.
- The interface ontology [Kultsova et al., 2017]: This ontology aims at modeling the different aspects of the UI.

5.3.3.2 Knowledge resource re-engineering

The reuse of knowledge resources involves their re-engineering into ontology, since the selected resources need to be modified in order to serve the intended purpose. To this end, the third phase within the MIDEF ontology life cycle concerns the re-engineering of knowledge resources including non-ontological and ontological resources. Therefore, for each of the knowledge resources, relevant non-ontological resources aspects along with ontological concepts are selected to be reused and incorporated into the MIDEF ontology.

5.3.4 Ontology design

The research conducted in this thesis focuses on three ontology modules to define the MIDEF ontology. As depicted in Figure 5.3, the MIDEF ontology is designed as a modular ontology composed of separate modules that covers different knowledge areas. To be more specific, the present ontology includes three modules, namely: "Design Pattern", "User Interface", and "User Profile" module. In order to achieve connections between these three modules, additional concepts that are sub-concepts of "Strategy" have been included in the ontology. First, the "Selection Strategy" concept is created to make a link between the "Design Pattern" and "User Profile" module. This concept defines the HCI design patterns that are considered relevant to solve the design problem related to the user needs or preferences associated with a certain user profile. Second, the "AdaptationStrategy" concept is defined to make a connection between the "Design Pattern" and "User Interface" module. This concept describes the relationship between the target UI and the selected HCI

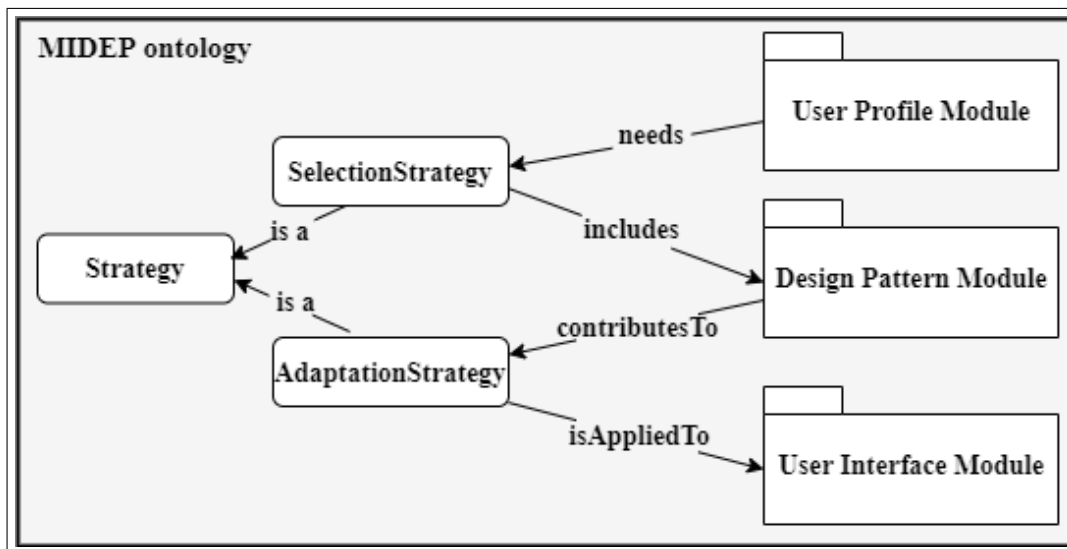


Figure 5.3: MIDE ontology modules

design patterns in order to enable the adaptation of the UI at run-time from the analysis of design solutions provided by the selected HCI design patterns.

The key features of the "Design Pattern" module, "User Interface" module, and "User Profile" module are further described below.

- **Design Pattern Module**

The "Design Pattern" module is the main module of the MIDE ontology and is intended to describe knowledge regarding HCI design patterns. Figure 5.4 depicts an excerpt of the conceptual model with the most relevant concepts and relationships related to the present module. The "DesignPattern", which is defined at the highest level of the conceptual model, represents the main concept of the present module. This concept is further specialized into "InteractionDP", "CustomizationDP", and "MiscellaneousDP" concepts. First, the "InteractionDP" concept represents a category of design patterns that provide design solutions related to the interface elements that are part of a UI. Second, the "CustomizationDP" concept consists of design patterns that offer design solutions related to the UI look and feel. Finally, the "MiscellaneousDP" concept refers to other design patterns that do not belong to the interaction and customization categories.

The principal aspects of the "DesignPattern" concept are characterized in terms of problem, solution, condition, and group. The information of these terms are modeled by the following concepts: "DesignPatternProblem", "Solution", "Condition", and "DesignPatternGroup". The "DesignPatternProblem" concept addresses the problem to be solved by a design pattern. In turn, the "Solution", "Condition", and "DesignPatternGroup" concepts address the solution provided by a design pattern, its context of use, and its group respectively. Table 5.5 reports the modeling role of each concept used to describe the "Design Pattern" module.

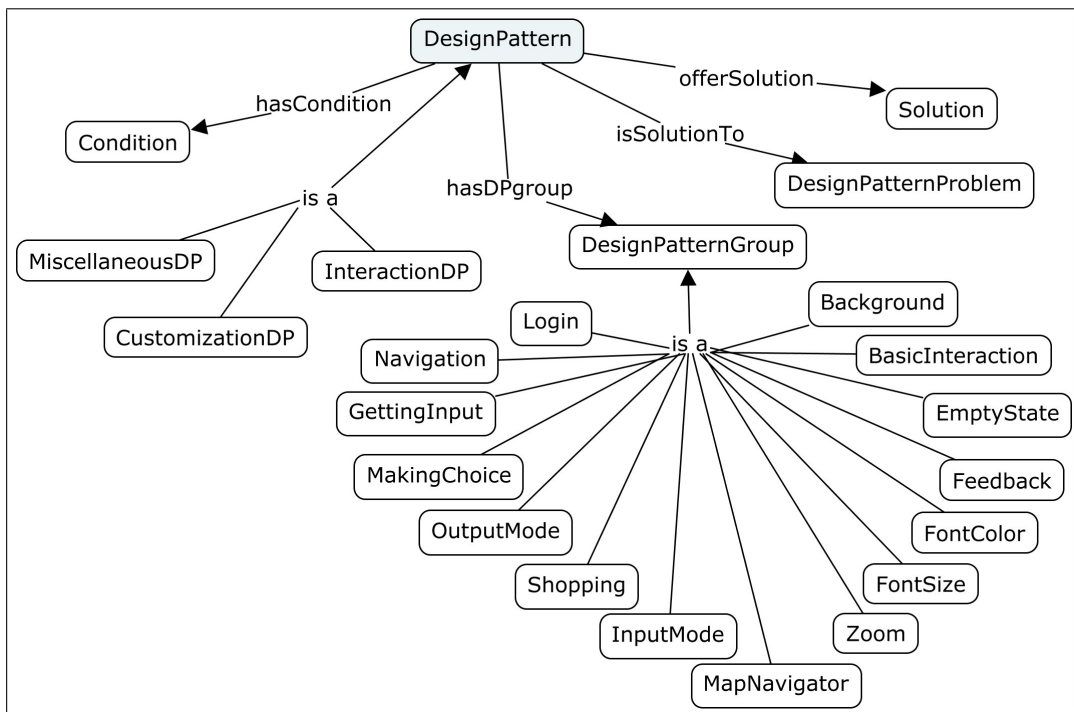


Figure 5.4: Conceptual model of the Design Pattern module

Table 5.5: Concepts dictionary of the Design Pattern module

Concept	Description
DesignPattern	This concept defines reusable solutions for UI design.
InteractionDP	This concept refers to a category of design pattern that offers design solutions related to UI elements
CustomizationDP	This concept refers to a category of design pattern that provides design solutions related to the look and feel of the UI.
MiscellaneousDP	This concept defines the miscellaneous category that refers to other design patterns, which could not be considered in the customization and interaction categories
DesignPatternProblem	This concept defines the design problem that can be solved by design patterns.
Solution	This concept describes the design solution proposed by design patterns for solving a specific design problem.
Condition	This concept refers to a set of information items that act as inputs for triggering design patterns
DesignPatternGroup	This concept represents the design pattern group, such as background, basic interaction, font size, or input mode.

• User Interface Module

The "User Interface" module covers the knowledge specializing in the "UserInterface" concept, which is the central concept of the present module. As shown in Figure 5.5, the "UserInterface" concept is linked to concepts "UIElement" and "UIElementType" with the relations "hasInterfaceElement" and "hasElementType", respectively. For instance, the "UIElement" is modeled using the following sub-concepts: "UIComponent" and "UIFeature". In turn, the "UIElementType" has three sub-concepts, namely "DisplayElement", "MultimediaElement", and "InteractionElement". The description of the main concepts presented in the "User Interface" module is provided in Table 5.6.

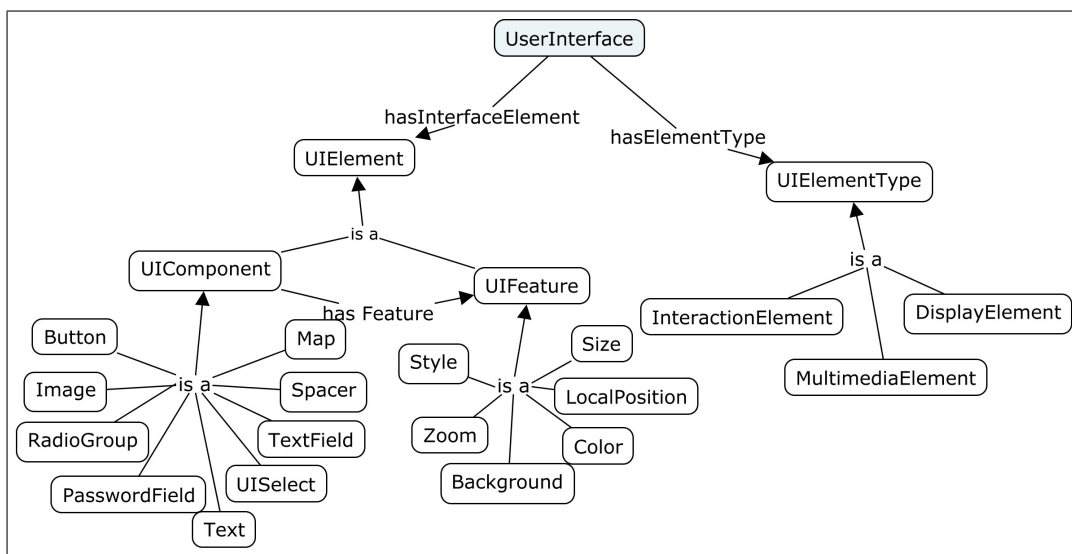


Figure 5.5: Conceptual model of the User Interface module

Table 5.6: Concepts dictionary of the User Interface module

Concept	Description
UserInterface	This concept represents a set of interface elements.
UIElement	This concept defines interface items that can be interface components or interface features.
UIElementType	This concept corresponds to the type of interface element, such as interaction element, multimedia element, or display element.
UIComponent	This concept refers to interactive interface components, such a button, image or spacer.
UIFeature	This concept refers to interface component settings, such as size, style, or color.

• User Profile Module

The "User Profile" module includes knowledge about a user such as his interest, preference, disability, and context. As illustrated in Figure 5.6, the central concept of the present module

is the "User" concept. This latter is semantically linked to a number of relevant concepts. For instance, "User" is linked to concepts "Context", "Activity", "Disability", "Goal", "Interest" with the relations "hasContext", "hasActivity", "hasDisability", "hasGoal", and "hasInterest", respectively. Table 5.7 summarizes the concepts used to define the "User Profile" module.

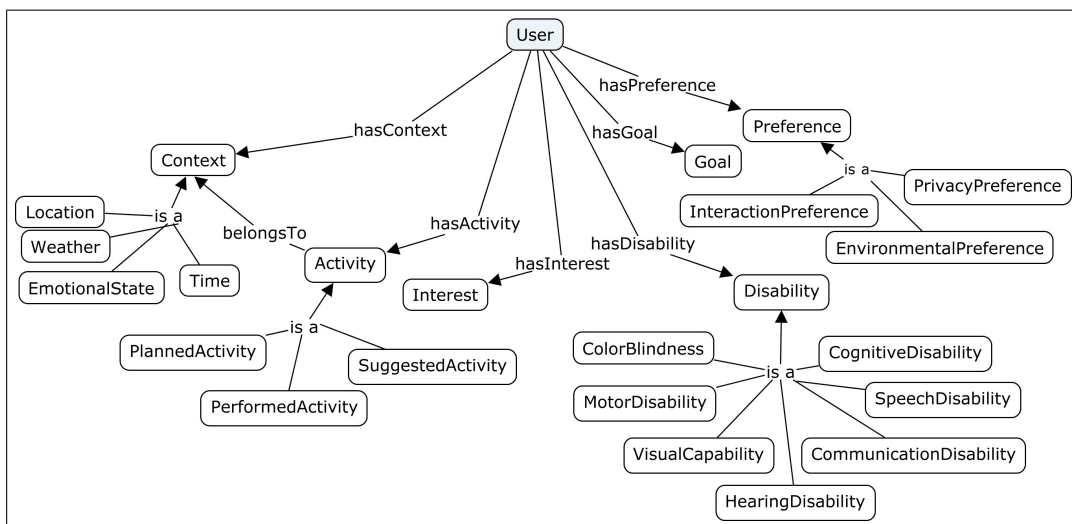


Figure 5.6: Conceptual model of the User Profile module

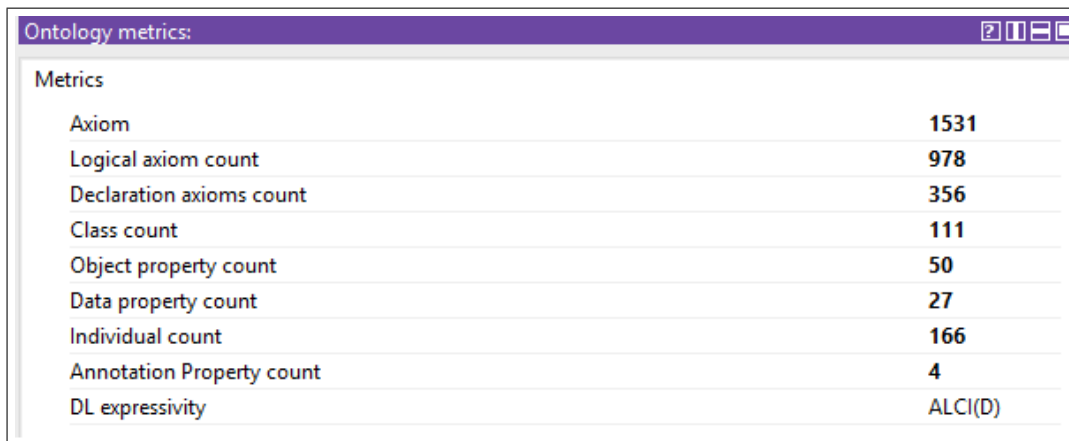
Table 5.7: Concepts dictionary of the "User Profile" module

Concept	Description
User	This concept defines the user who has a profile interacting with the UI.
Context	This concept refers to the information that characterizes the environment surrounding a user, for example weather, location, time or emotional features.
Activity	This concept captures information regarding the user's daily activity, for example planned activity, performed activity, or suggested activity.
Interest	This concept describes any existing information that the user may find interesting.
Disability	This concept represents a subset of impairments that can affect the user.
Goal	This concept refers to the targeted goal of the user.
Preference	This concept defines information regarding users' preferences, such as privacy preference, environmental preference, or interaction preference.

5.3.5 Ontology implementation

This phase refers to the transformation of the formal and well-founded conceptual model, resulted from the previous phase, into a computable model. For the purpose of providing a computable model, the implementation of the MIDEP ontology is conducted in OWL language by using the Protégé tool. The selection of OWL language for explicitly representing knowledge in the ontology will provide expressive and powerful representations. Moreover, the choice of Protégé tool is based on the fact that it is the most accepted and widely used open-source tool that allows intuitive construction of ontologies.

At this stage, the MIDEP ontology's classes, properties, and instances are created using Protégé. Annotations are also included in the present ontology. The resulting ontology's main characteristics are summarized in Figure 5.7. In particular, the developed ontology entails a total of 1531 axioms, 111 classes, 50 object properties, and 27 data properties. The resulting ontology file is accessible through the MIDEP ontology URI.



Ontology metrics:	
Metrics	
Axiom	1531
Logical axiom count	978
Declaration axioms count	356
Class count	111
Object property count	50
Data property count	27
Individual count	166
Annotation Property count	4
DL expressivity	ALCI(D)

Figure 5.7: Main characteristics of the MIDEP ontology

Ensuring modularity in the MIDEP ontology is initiated from the first phase; thus, separate ontology modules are developed to reinforce reusability. Subsequently, all Classes and properties for each module are defined with their corresponding prefixes. In particular, the namespace of the "Design Pattern" module is represented by the prefix "DP", while the "User Interface" and "User Profile" modules are represented by the prefix "UI" and "UP", respectively. The hierarchical fragment, taken from Protégé, of key classes and subclasses within the MIDEP ontology is shown in Figure 5.8. For example, the "DP:DesignPattern" class has "DP:CustomizationDP", "DP:InteractionDP", and "DP:MiscellaneousDP" sub-classes.

In order to make the structure of each concept, object properties and data properties are created. While object properties refers to the connection between individuals, data properties aim to achieve a link between the concept individual and the data literals. Figure 5.9 (a) and Figure 5.9 (b) illustrate a hierarchical view of object property and data property.

Instantiating the ontology's main concepts with individuals is also performed in the present phase. To this purpose, individuals, which act as instances of classes and sub-classes, are created. For example, the "DP:DesignPattern" class includes a total of 45 HCI design patterns. Figure 5.9 (c) shows an instantiation of the "DP:DesignPattern" class. An excerpt of our catalog of HCI design patterns is provided in Appendix 9.5.

5.4 Concluding Remarks

This chapter has summarized an ontology method for the specification of HCI design patterns using the NeOn methodology. It outlines the method for the initiation, reuse, re-engineering, design, and implementation of the MIDEP ontology.

The next two chapters consider the developed MIDEP ontology for recommending the most relevant HCI design patterns (Chapter 6) and for generating UIs for the target

application (Chapter 7). The present ontology is further assessed and validated according to different evaluation approaches as described in Chapter 8.

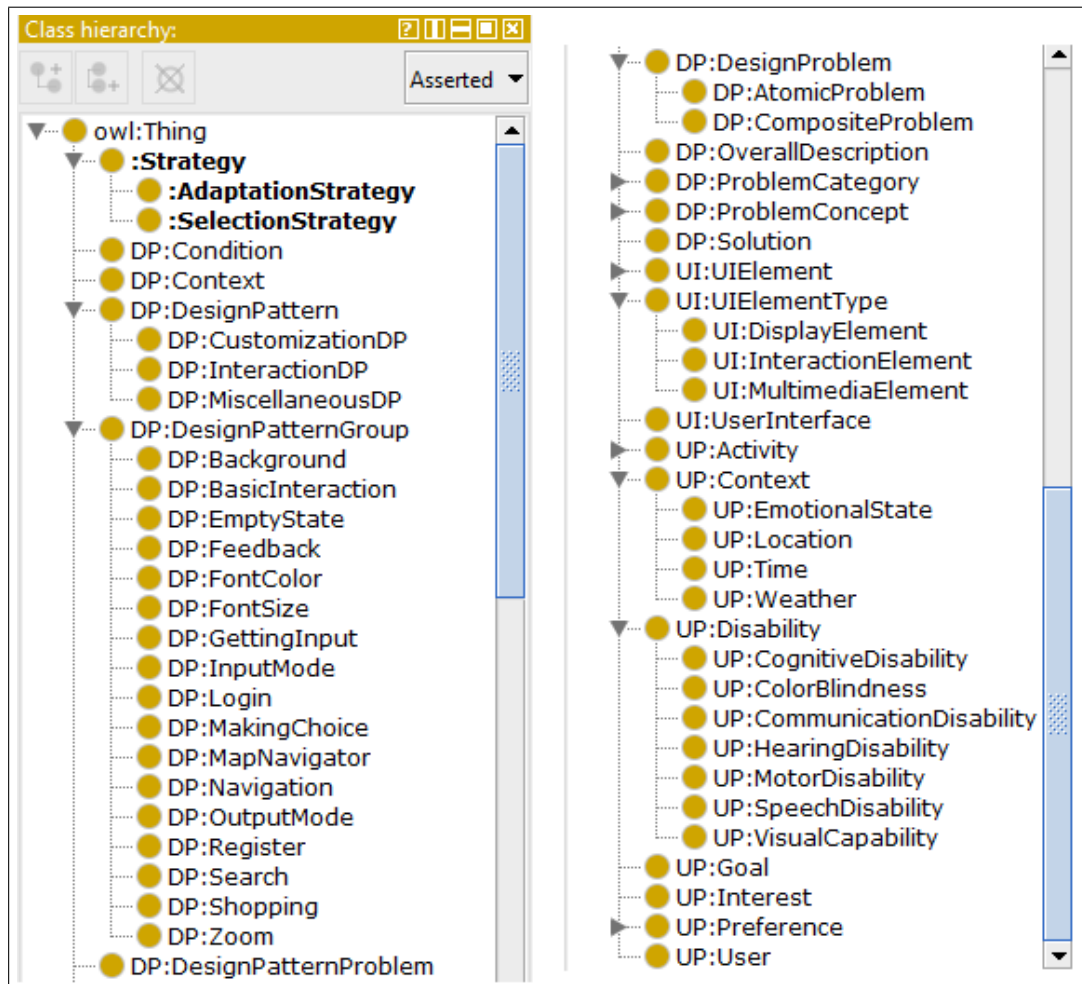


Figure 5.8: Screenshot of OWL ontology classes implemented using Protégé

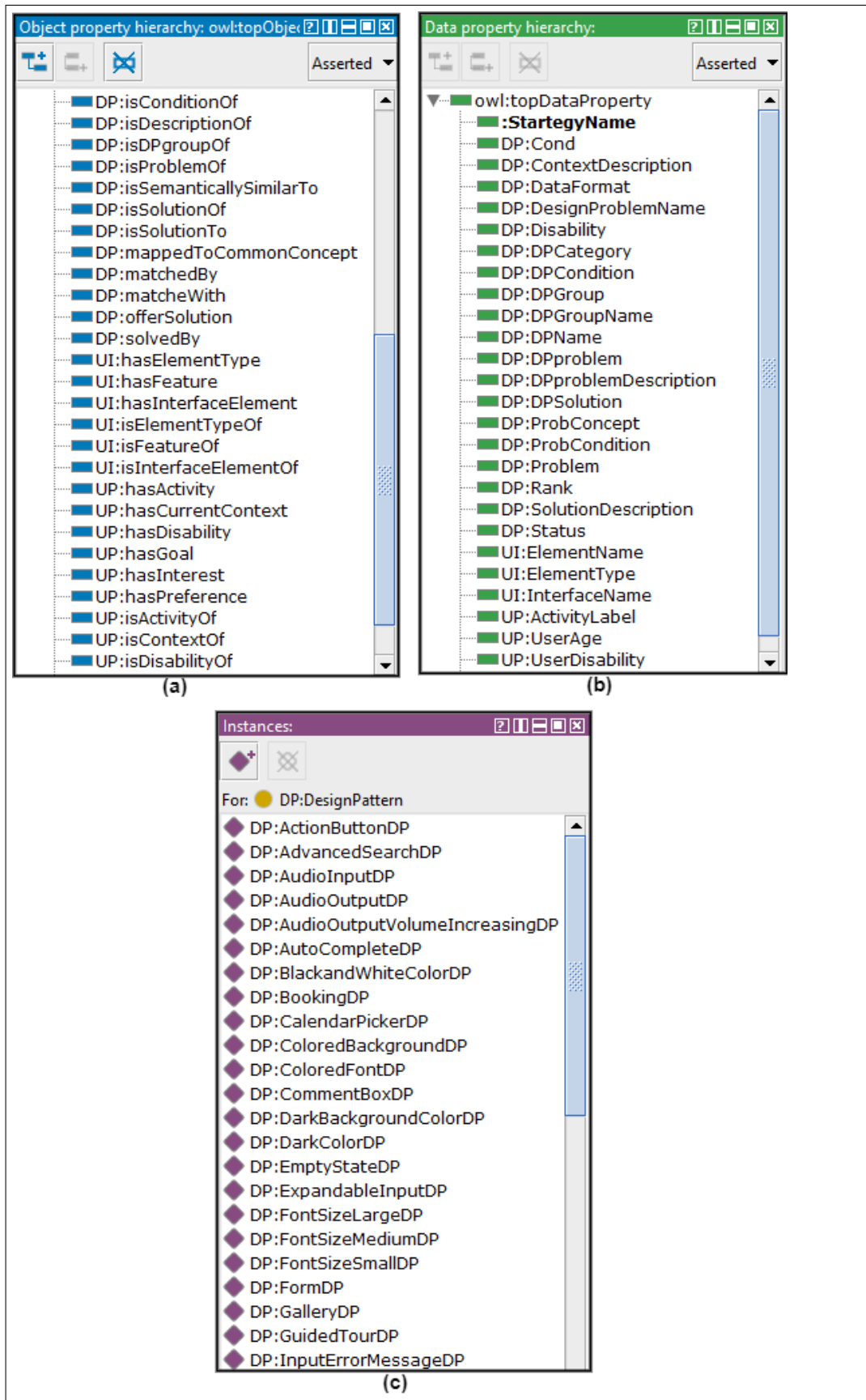


Figure 5.9: Screenshot of OWL ontology object properties, data properties, and instances implemented using Protégé

Design Pattern Recommender System

6.1 Introduction

In this thesis, we propose an automatic recommender system, named IDEPAR, which is the first system within the global AUIDP framework introduced in Chapter 4. This system aims to recommend the most relevant design patterns to solve a given design problem and to help designers and developers in the HCI field.

The remainder of this chapter is structured as follows: In Section 6.2, an overview of the IDEPAR system is provided, after which an illustration of each module that composes the system's architecture and its implementation features are described in Section 6.3 and Section 6.4, respectively. Finally, Section 6.4 gives some concluding remarks.

6.2 Recommender System Overview

The proposed IDEPAR system is based on a hybrid approach that relies on a mixed combination of two techniques considered in recommendation systems. The focus of the present approach is to produce relevant recommendations by using text-based and ontology-based techniques. The overview of the present IDEPAR system, depicted in Figure 6.1, reflects the hybrid recommendation approach; in particular, the NLP module considers the text-based technique, and the semantic module relies on the ontology-based technique. The input to the recommender system is the description of design problems that could be full-text or keywords, while the output is the most relevant HCI design patterns.

6.3 Recommender System Architecture

The IDEPAR system entails strategies to deal with design pattern recommendations by supporting a hybrid recommendation approach. As depicted in Figure 6.2, the IDEPAR system includes two main modules that interact among them, including the NLP module

6. DESIGN PATTERN RECOMMENDER SYSTEM

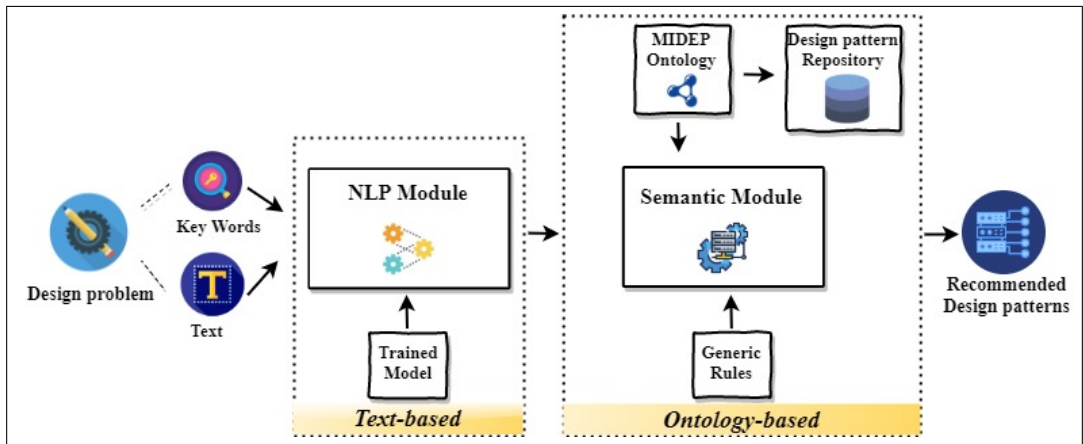


Figure 6.1: Overview of the IDEPAR system

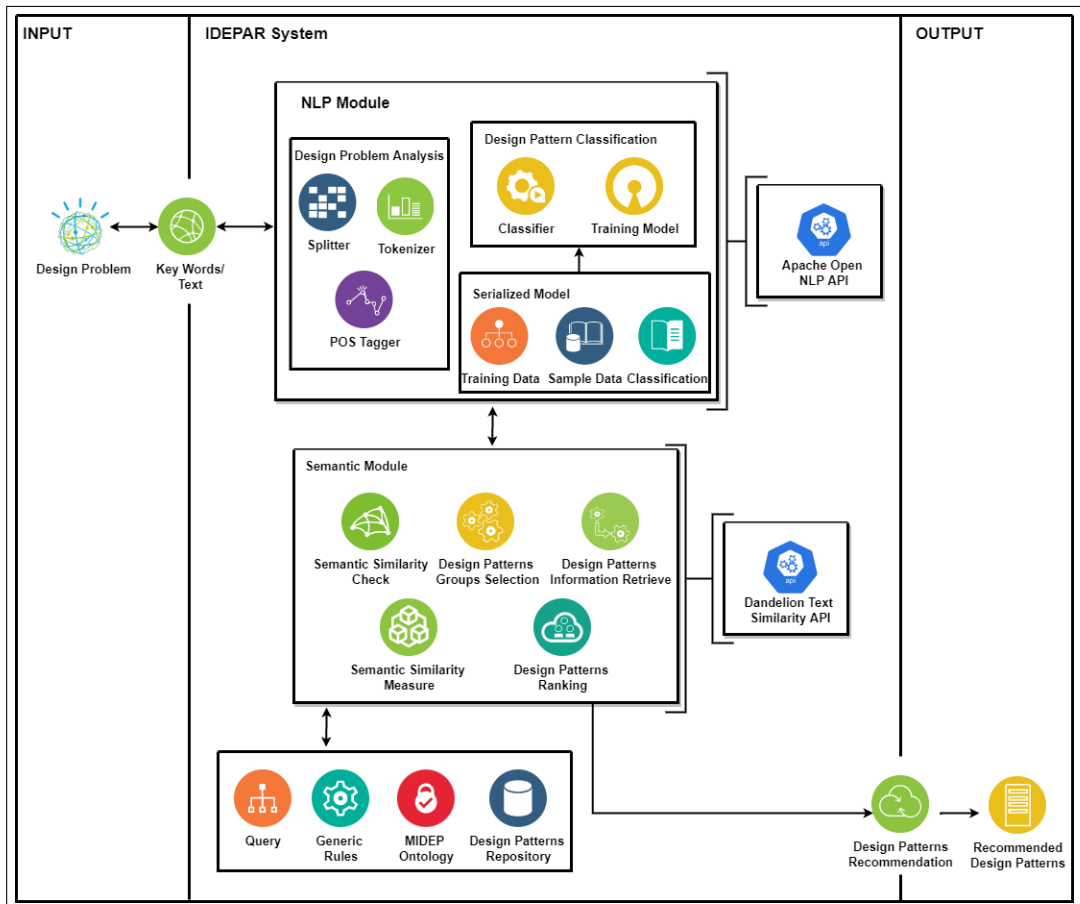


Figure 6.2: Architecture of the IDEPAR system

and the semantic module. A brief description of each module is introduced in the following subsections.

6.3.1 NLP module

The purpose of the NLP module is to deal with the description of the given design problem, presented in natural language, based on text preprocessing methods. While these methods cannot directly be considered for the recommendation goal, the NLP module is a fundamental component within the IDEPAR system to prepare and transform the design problem into a more accessible form for the semantic module. Figure 6.3 depicts the processing components considered in the NLP module, including the design problem analysis and design problem classification component. The first component is used to analyze problem scenarios syntactically, while the second one is used to affect categories for each atomic problem. A description of each component is provided below.

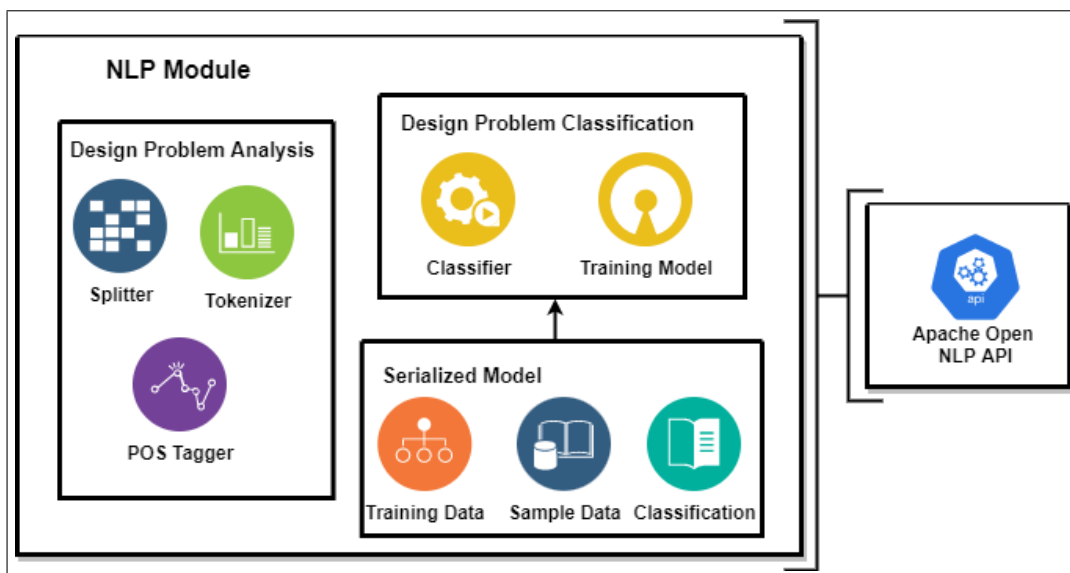


Figure 6.3: NLP module overview

- The design problem analysis:** The design problem analysis is the first component in the NLP module and is applied to the description of design problems given by designers or developers. As illustrated in Figure 6.3, this component supports the standard information retrieval methods including sentence splitter, tokenization, and Part Of Speech (POS) tagging strategies. The aim of these strategies is to transform raw text into fragments while eliminating irrelevant information. More specifically, the first strategy is used to split composite design problems into atomic ones. The obtained atomic design problems are applied in the second strategy and transformed into small textual fragments called tokens. Finally, the resulting tokens are annotated using the POS tagging strategy.
- The design problem classification component:** The main objective of this component is to deliver, for each atomic design problem identified in the first component, a category name allowing the classification of design problems into predefined categories. The present component deals with design problem classification based on NLP auto-categorization. To this purpose, a training model is considered to predict the appropriate category for each atomic design problem. The training model is

generated from a set of training data that serve as a source of knowledge for the classification component to make decisions about design problem categories.

6.3.2 Semantic module

The semantic module, outlined in Figure 6.4, is another crucial module within the IDEPAR system that provides HCI design patterns recommendations. Its input data are design problem categories affected in the NLP module, which are used to develop the knowledge base. This module mainly operates on a semantic knowledge base, which involves ontologies for representing design problems and HCI design patterns. In particular, the extraction of relevant design patterns extends the use of the semantic knowledge base where the knowledge derives the IDEPAR system’s recommendation. A detailed description of the semantic knowledge base along with the recommendation process provided by the semantic module are presented in the following subsections.

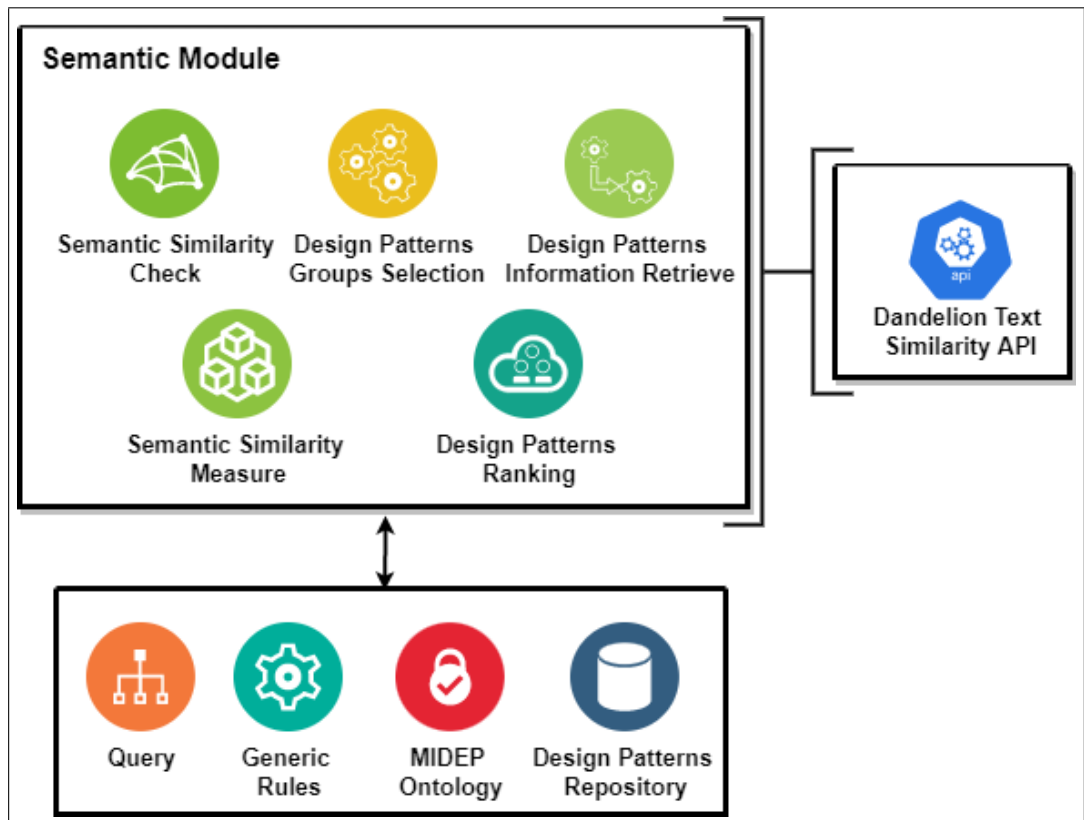


Figure 6.4: Semantic module overview

6.3.2.1 Semantic knowledge base

Ontologies are fundamental for the semantic module serving as means: First, for modeling design patterns along with design problems. Then, for the definition, the storage and the retrieval of the semantic knowledge base. Finally, for the generation of relevant recommendations. In the present module, the semantic knowledge base considers ontologies in OWL as a knowledge representation language. Knowledge is retrieved via SPARQL

queries as ontology retrieving language. The conceptual foundation of the knowledge base within the IDEPAR system is set by the following ontologies:

- **Design pattern ontology:** This ontology is a formal specification of design patterns in the HCI domain and it is one of the main modules of the MIDEP ontology presented in Chapter 5. In this knowledge base, a collection of 45 HCI design patterns are considered. This collection is given in the form of instances of the design pattern ontology and used for the recommendations provided by the IDEPAR system.
- **Design problem ontology:** The target of this ontology is to provide a structured description of design problems. In this work, design problems are defined as problems that reflect the requirements of designers and developers while developing a specific UI. They also refer to issues faced by end users, including interface display, navigation, look and feel etc. Subsequently, the present ontology supports properties describing the design problem, such as “Name”, “DataFormat”, and “Status”. Moreover, this ontology provides other concepts that are related to the design problem concept as illustrated in Figure 6.5.

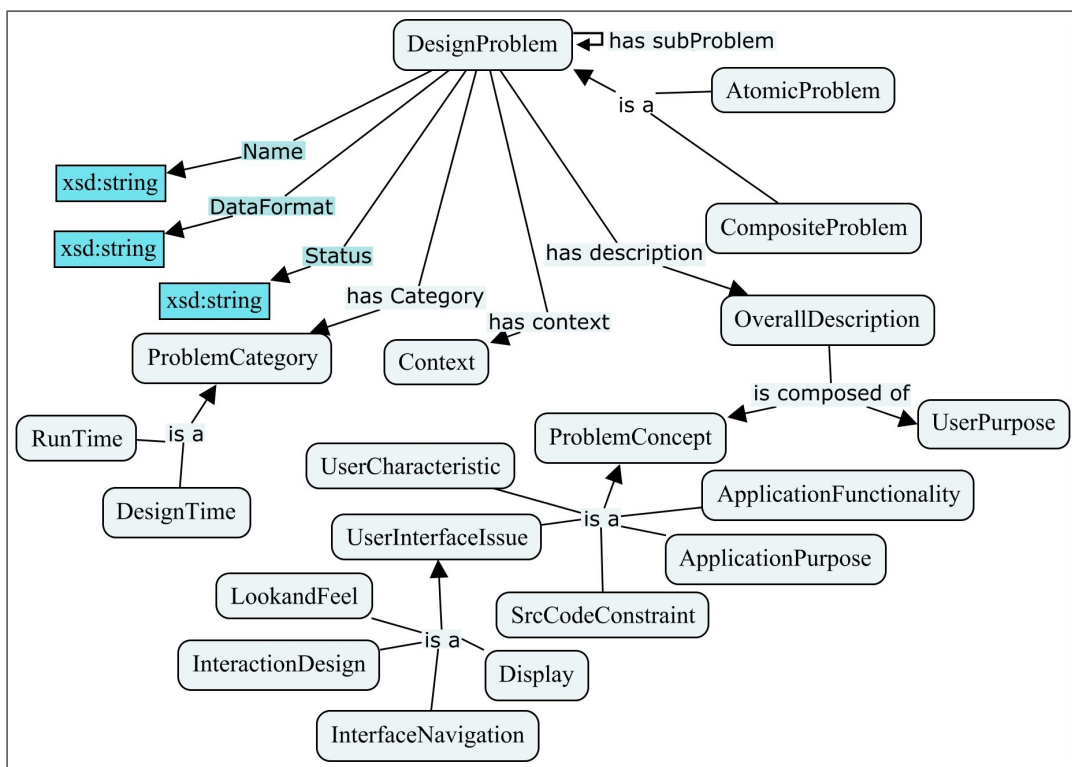


Figure 6.5: Design problem ontology model

6.3.2.2 Recommendation process

The semantic module deals with the identification and the selection of the most relevant HCI design patterns to make them generated by the IDEPAR system and thus accessible for designers and developers. To this purpose, the semantic module implements a recommendation process composed of five sequential phases.

- **Semantic similarity check:** The main objective of this phase is to compute the semantic similarity between the given design problems and design patterns stored in the knowledge base. Based on the obtained Semantic Similarity Measures (SSM), the semantic module updates the ontology models by creating relations between concepts that are semantically similar. This phase is illustrated in Figure 6.6.

```

1  input: design-problem x, design-pattern-group y
2    Cx = getConcept(x)
3    Cy = getConcept(y)
4    do
5      SSM(Cx,Cy) = semanticSimilarity(Cx,Cy)
6      if SSM(Cx,Cy) then
7        createObjectproperty(Cx, isSemanticallySimilarTo, Cy)
8      else
9        Cz= newIndividual()
10       createObjectproperty(Cx, isSemanticallySimilarTo, Cz)
11     fi
12   until no more design pattern groups
13   return

```

Figure 6.6: Semantic Similarity Check Algorithm

- **Design patterns group selection:** the second phase in the semantic module concerns the selection of design patterns' group. To this end, the semantic module creates matching between design patterns' groups and design problem concepts by applying a set of inference rules. Using the inference results, the semantic module query the ontology models to retrieve the list of design patterns' groups through SPARQL queries. An inference rule example in textual and Jena format is illustrated in Table 6.1.
- **Design patterns information retrieve:** the purpose of this phase is to select an initial set of HCI design patterns using the group identified in the previous phase. To this end, the semantic module infers the design pattern ontology using a reasoning mechanism based on the object property "hasDPGroup".
- **Semantic similarity measure:** After retrieving an initial list of HCI design patterns, the semantic module is in charge of computing the semantic similarity between the design problem categories, affected in the first module, and the descriptions of design patterns that correspond to the selected design patterns groups.
- **Design patterns ranking:** The last phase in the semantic module aims to refine the initial list of design patterns using the SSM obtained in the previous stage. As a

result, the most relevant design patterns for the given design problem scenario are recommended based on the following Equation:

$$S(A,B) > \alpha \quad (1)$$

Where, A is the design problem category, B is the design pattern condition, and α is a threshold value for the SSM. As part of the design patterns ranking phase, it was noted that HCI design patterns with a SSM value below 0.4 were not relevant to the given design problems. Thus, a threshold of 0.4 is considered.

Table 6.1: A rule example for matching design pattern groups.

Textual Rule Format

Design Problem “x” is composed of Problem Concept “Conceptx”, Design pattern Group “y” has a context “Contexty”,and “Conceptx” is semantically similar to “Contexty”,
Then “y” matche with “x”.

Jena Rule Format

```
[MatchingRule:
  (?x rdf:type uni:DesignProblem)
  (?x uni:isComposedOfProblemConcept ?Conceptx)
  (?y rdf:type uni:DesignPatternGroup)
  (?y uni:hasContext ?Contexty)
  (?Contexty uni:mappedToCommonConcept ?Conceptx)
  ->
  (?y uni:matcheWith ?x)
]
```

6.4 IDEPAR Implementation

The proposed IDEPAR system has been implemented in a server-side and client-side architecture as illustrated in Figure 6.7. While the server-side considers the development of services provided by the IDEPAR’s modules including the NLP module and the semantic module, the client-side concerns the visualization and the retrieval of the recommended HCI design patterns for a given design problem.

6.4.1 Server-side implementation

The server-side of the IDEPAR system is implemented as a Web service, which could be operated using RESTful API. Figure 6.7 depicts the micro-services running on the server-side to provide the services of the IDEPAR’s main modules to the client-side. In particular, the environment used for the server-side implementation is Eclipse IDE using Java programming language and Spring framework. Moreover, other tools and technologies are considered, including Apache Maven to build the Java project and for dependency management, Jersey to provide RESTful Web services, Apache Tomcat to host Jersey and

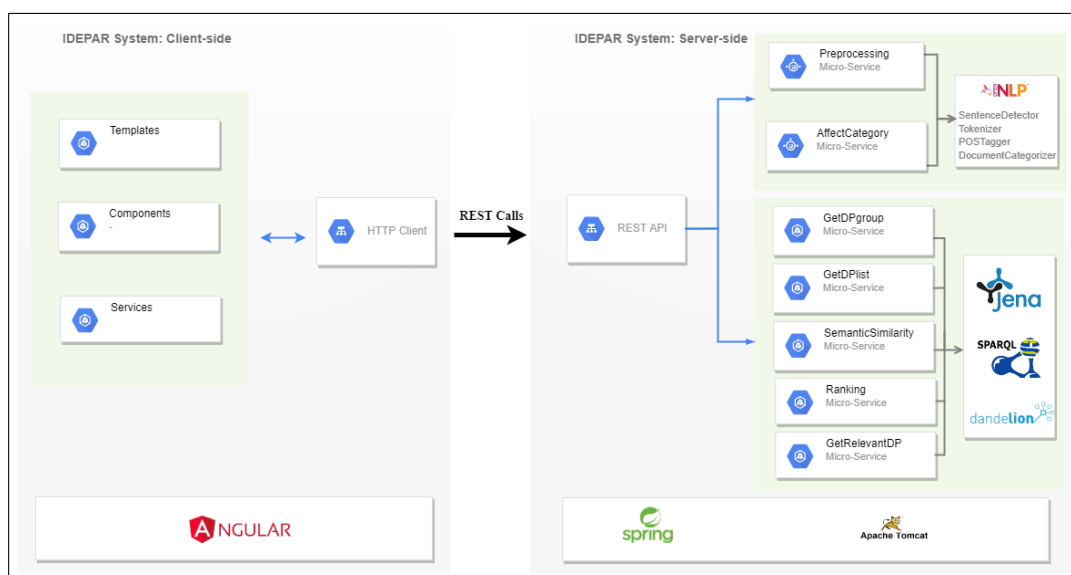


Figure 6.7: IDEPAR system: server-side and client-side implementation

RESTful Web services, and Apache Jena for reasoning over ontologies, presented in the semantic knowledge base of the IDEPAR system, and for processing SPARQL queries. Finally, Apache OpenNLP API and Dandelion API are used to process natural language text and to identify semantics between texts, respectively.

6.4.2 Client-side implementation

Besides the server-side implementation, we propose the client-side implementation. This implementation is performed using the AngularJS framework that is considered to build the client-side of the AUIDP framework as a Web application that calls the REST web services provided by the server-side. Finally, the development of the present application is built with Visual Studio Code IDE using JavaScript, HTML and Sass programming languages.

6.4.3 Design pattern recommendation example

In order to illustrate how the proposed IDEPAR system is applied for a particular design problem scenario, we provide in this subsection a HCI design pattern recommendation example. As an example of a design problem, the following scenario is considered: "The user cannot perceive colors, The user needs to find the location of a point of interest" (DPS-1). A detailed description of the outputs of each module within the IDEPAR system is further presented.

The given design problem was processed through four steps of the NLP module. First, in the splitter step, DPS-1 was divided into sentences using the Sentence Detection API that allows to extract sentences from the scenario. In this step, long sentences were split into short sentences in order to identify atomic design problems. As shown in Figure 6.8, DPS-1 was divided into two atomic problems: "the user cannot perceive colors" (DPS-1-1) and "The user needs to find the location of a point of interest" (DPS-1-2).

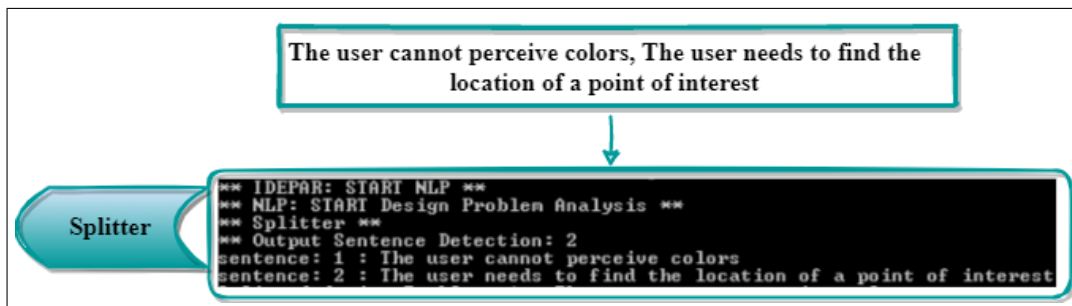


Figure 6.8: Splitter results for DPS-1

Then, in the second step of the NLP module, atomic design problems, including DPS-1-1 and DPS-1-2, were tokenized using the Tokenizer API. They were transformed into objects where each word was represented as a small fragment, called a token. Next, in the third step, the NLP module assigned POS tags to tokens, which are obtained from the tokenizer step. The results of the tokenizer and POS tagger step are depicted in Figures 6.9 and 6.10, respectively.

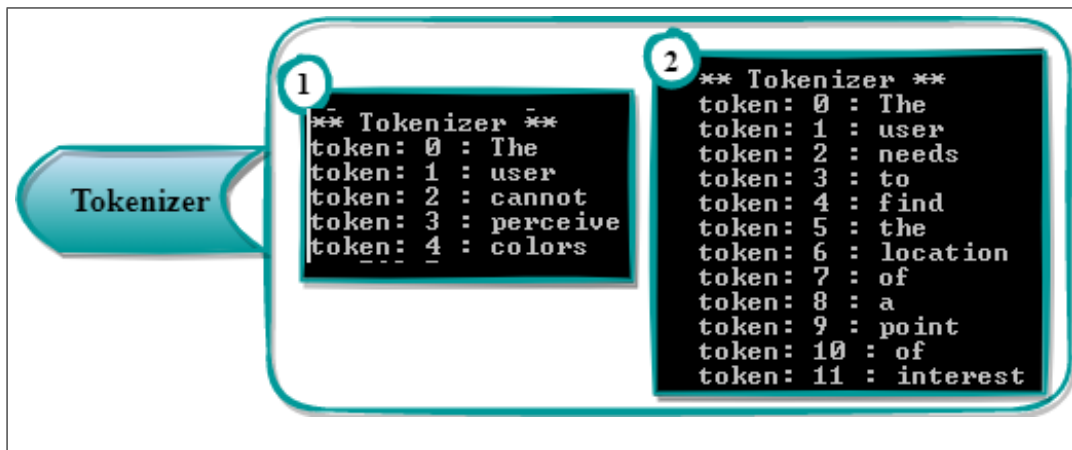


Figure 6.9: Tokenizer results for DPS-1

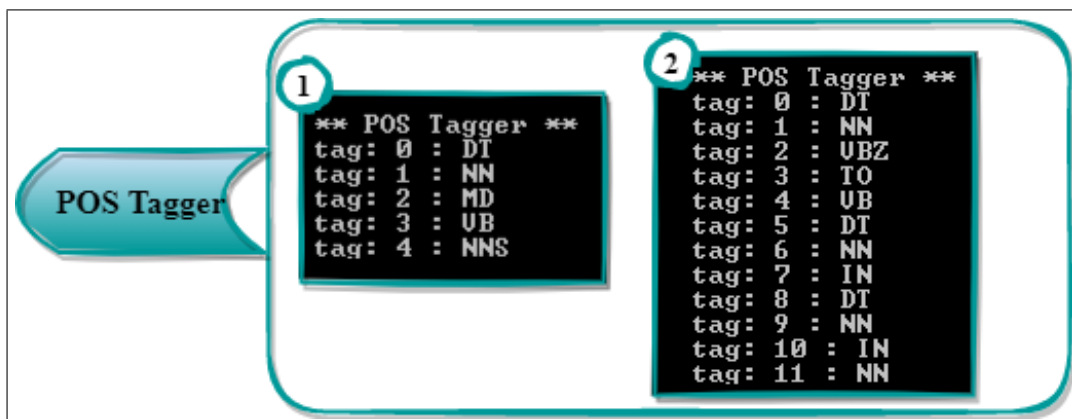


Figure 6.10: POS tagger results for DPS-1

Finally, using the tags assigned in the POS tagger step, only nouns and verbs were part

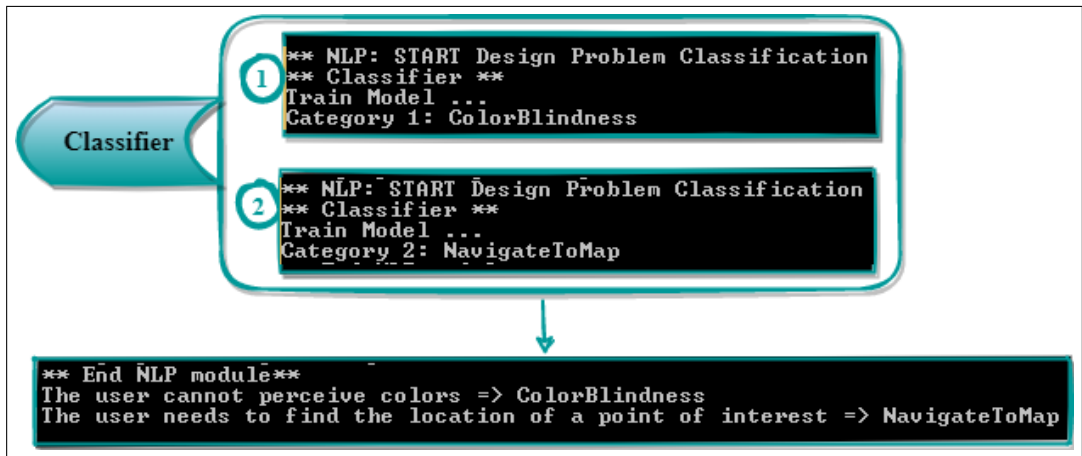


Figure 6.11: Classifier results for DPS-1

of the classifier step. The Document Categorizer API was considered in this step to affect categories for each atomic problem. Particularly, a training model was used to identify the appropriate problem categories, taking as input the nouns and verbs of each atomic problem. As shown in Figure 6.11, the categories "Colorblindness" and "NavigateToMap" were assigned to DPS-1-1 and DPS-1-2, respectively.

The result of the NLP module for DPS-1 is illustrated in Figure 6.12. The given design problem was passed as input parameters to the "getNLPmoduleResult" service that communicates with the "Preprocessing" and "AffectCategory" micro-services previously presented in Figure 6.12. The response body of the "getNLPmoduleResult" service was represented in a string format (Atomic design problem => Category) that would be passed into the semantic module.

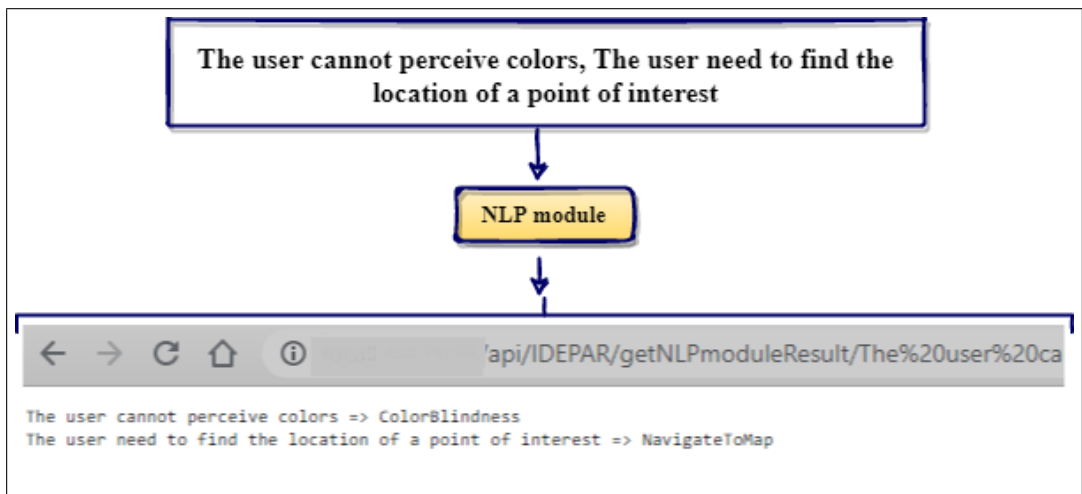


Figure 6.12: NLP module results for DPS-1

The obtained results from the NLP module were used in the semantic module to select the most relevant HCI design patterns for the given design problem. In this sense, the problem categories DPS-1-1 and DPS-1-2 were passed as input parameters to the "getSemantic-moduleResult" service, which communicates with the micro-services shown

in Figure 6.7. The response body of this service was provided in JSON response format. An excerpt of the recommended HCI design patterns selected by the semantic module to solve the given design problem (DPS-1) is depicted in Figure 6.13.

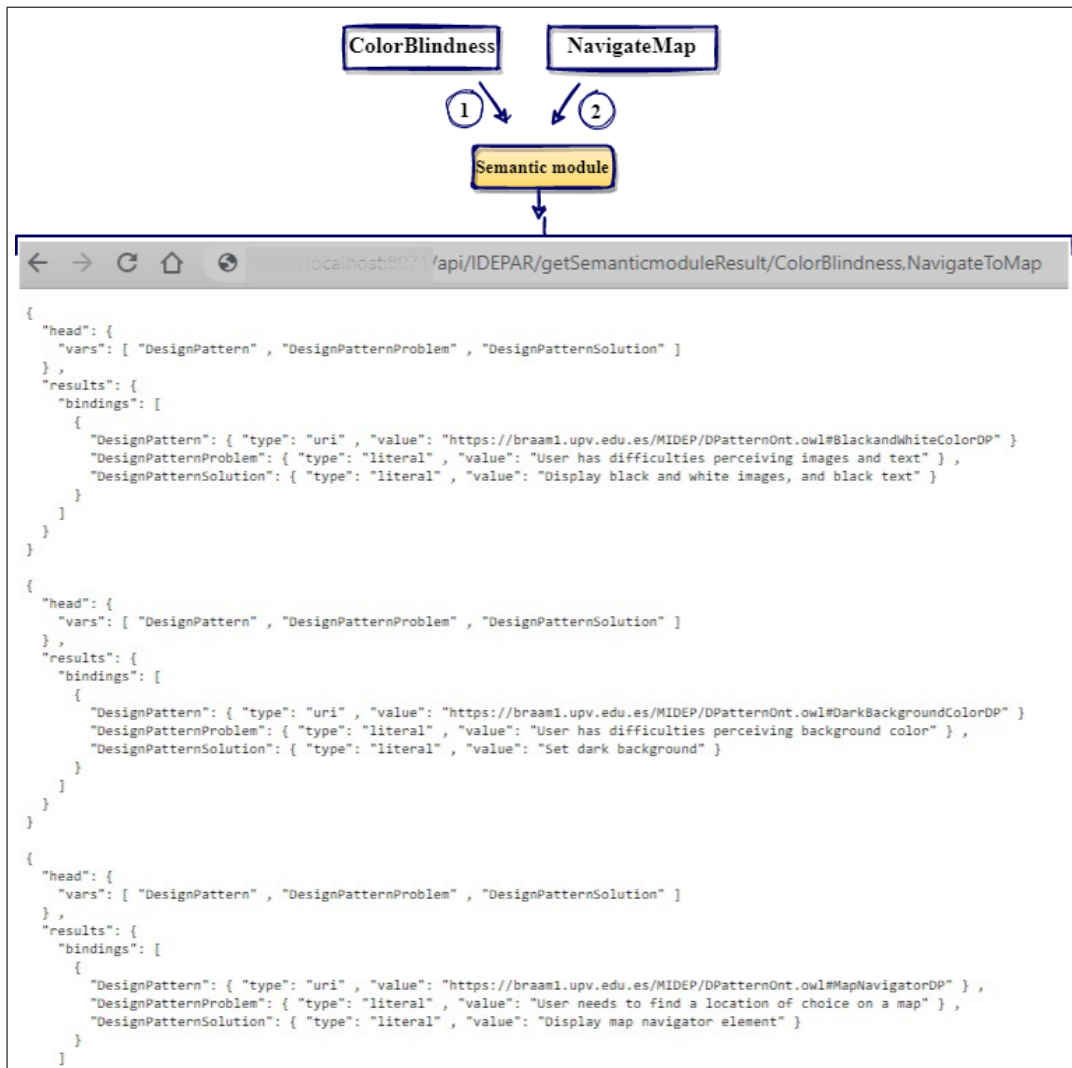


Figure 6.13: Semantic module results for DPS-1

6.4.4 IDEPAR prototype

In this subsection, we present a Web application that allows displaying the recommended HCI design patterns for a given design problem. In order to process the design pattern recommendation requests received from developers and designers, the present Web application communicates with the REST Web services provided by the IDEPAR system. This application was developed using Spring Boot, Angular, and other technologies. The main interface of the developed prototype, outlined in Figure 6.14, presents an overview of the IDEPAR system and provides two options to access different interfaces: the first interface shows the repository of HCI design patterns described in Chapter 5, whereas the second interface displays design patterns selected by the IDEPAR system for particular design problems.

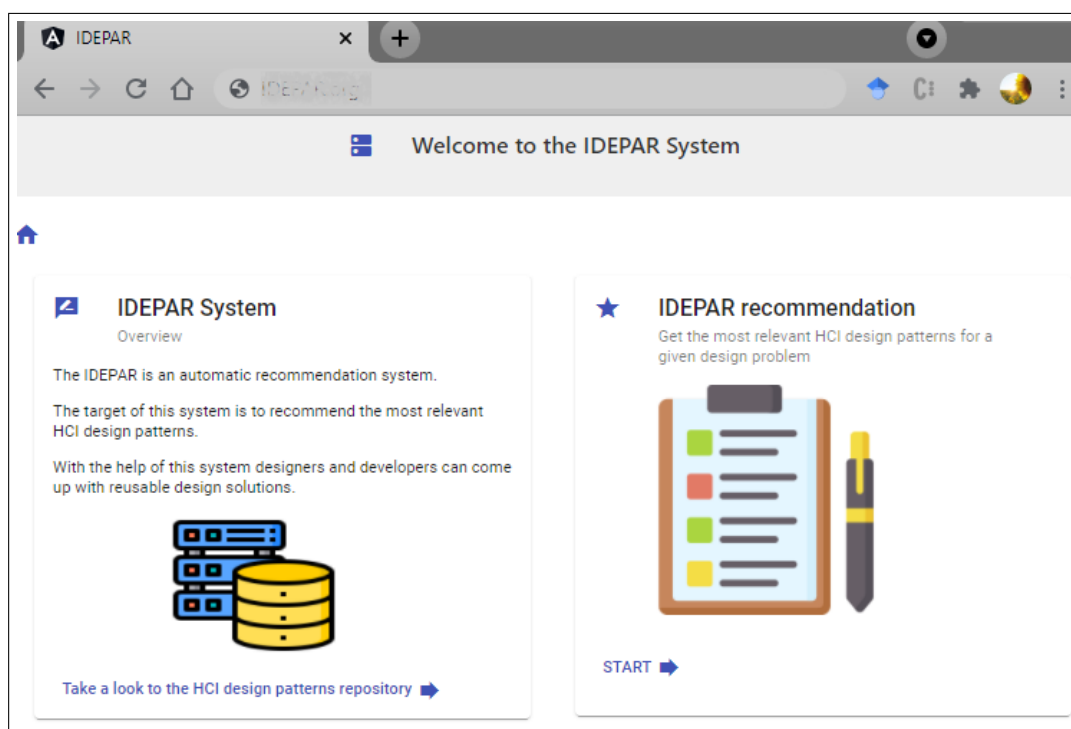


Figure 6.14: IDEPAR application main interface

To illustrate the accomplishment of the proposed IDEPAR system regarding various design problem scenarios, we considered the following two scenarios :

- **DPS-1:** The user cannot perceive colors, The user needs to find the location of a point of interest.
- **DPS-2:** The user is walking, The user needs to connect with an application.

First, to deal with DPS-1, Figure 6.15 shows the interface for selecting DPS-1 using the user characteristic and user purpose options. After submitting the design problem, the list of the recommended HCI design patterns is displayed. In particular, five HCI design patterns were recommended by the IDEPAR system to solve DPS-1 as shown in Figure 6.16.

Second, to solve the design problem (DPS-2), Figure 6.17 presents the interface to submit the problem scenario using the user characteristic and user purpose options. In this case, four design patterns were selected by the IDEPAR system as shown in Figure 6.18. Each design pattern item is displayed with its name and problem. Additional information regarding the recommended design patterns, including their corresponding group and design solution, could be displayed by clicking on one of the design pattern items (e.g., AudioInput).

1 Design Problem Scenarios — IDEPAR Design Patterns recommendation — 3 Finish

– Choose your design problem

Design Problems

User Profile ^

User characteristic
The user cannot perceive colors v

User purpose
The user needs to find the location of a point of interest v

Interface Issue v

➔ Submit data and get the Recommended Design patterns using IDEPAR

Figure 6.15: Selection of design problem DPS-1

1 Design Problem Scenarios — 2 IDEPAR Design Patterns recommendation — 3 Finish

Design problem: The user needs to find the location of a point of interest –The user cannot perceive colors–

– Rate the relevance of the recommended design patterns

Nbr	DesignPatternName	DesignPatternProblem
1	ZoomIn	Different users need different zoom level
2	ZoomOut	Different users need different zoom level
3	MapNavigator	User needs to find a location of choice on a map
4	DarkBackgroundColor	User has difficulties perceiving background color
5	WhiteColor	User has difficulties perceiving images and text

< Back > Next

Figure 6.16: List of recommended design patterns for DPS-1

6. DESIGN PATTERN RECOMMENDER SYSTEM

Figure 6.17: Selection of design problem DPS-2

Nbr	DesignPatternName	DesignPatternProblem
1	Login	The user needs to get access and identify themselves
2	Audiolnput	The user Cannot use interface element to input data

Figure 6.18: List of recommended design patterns for DPS-2

6.5 Concluding Remarks

This chapter has presented a system, named IDEPAR, for recommending the most relevant HCI design patterns. The proposed IDEPAR system is based on a hybrid approach combining both text-based and ontology-based techniques with focus on supporting the automatic recommendation of design patterns for a given design problem. It entails two principal modules: (i) the NLP module, where the given design problems are preprocessed using

text-based technique, and (ii) the semantic module which involves the use of the semantic knowledge base to derive the system's recommendation.

At the end of this chapter, we have focused on describing the implementation of the IDEPAR system. As a proof-of-concept implementation, we have presented a recommendation example to highlight the outputs of the NLP and the semantic modules. Additionally, in order to process the design pattern recommendation requests received from developers and designers, we have developed a Web application that interacts with the proposed IDEPAR system.

User Interface Generator System

7.1 Introduction

The present chapter introduces an interface generator system, called ICGDEP that aims to generate the UI source code for Web and mobile applications. The proposed ICGDEP, which is the second system within the global AUIDP framework, relies on the use of HCI design that are recommended by the IDEPAR system.

The remainder of this chapter is organized as follows: Section 7.2 provides an overview of the general architecture of the ICGDEP system. Section 7.3 includes a description of the implementation of the ICGDEP system and shows a running example. Finally, Section 7.4 ends up the present chapter with a brief summary and a discussion.

7.2 System Architecture

In order to support the use of the HCI design patterns that are recommended by the IDEPAR system, previously introduced in Chapter 6, the aim of the ICGDEP system is to automatically generate the final UI source code. This generation is focused on Web and mobile applications as target applications. Therefore, the ICGDEP system especially relies on design patterns containing design solutions specifically used to generate UIs of the target applications.

Figure 7.1 depicts the overall architecture of the proposed ICGDEP system. It consists of three main components, namely pattern instantiation, pattern integration, and UI generation. The pattern instantiation component takes as input the list of the recommended design patterns to build pattern fragments. These fragments are then delegated to the pattern integration component to create the corresponding UI models. Based on these models, the UI generation component automatically generates the UI source code. The generated code is injected into the final UI of the target application by using the global AUIDP framework to support UI adaptation and generation at run-time. A detailed description of each component that constitutes the ICGDEP system is provided in the following subsections.

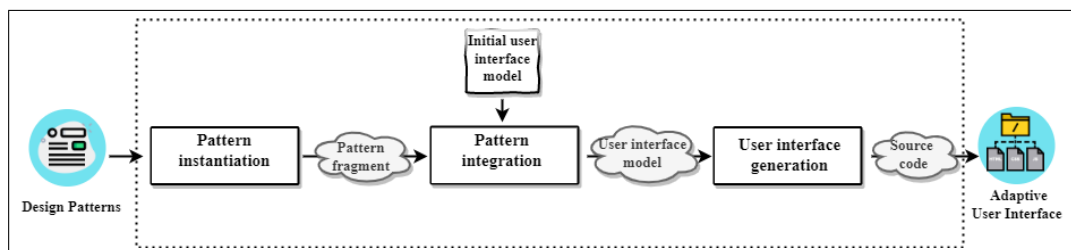


Figure 7.1: ICGDEP system architecture

7.2.1 Pattern instantiation

Once HCI design patterns are retrieved from the IDEPAR system, the pattern instantiation component provides support for their instantiation to allow their application in the ICGDEP system. Due to the fact that design patterns provide abstracted and generalized design solutions to recurring design problems, their application requires the concretization and specialization of the design solution provided by the design patterns. Consequently, the present component relies on a concretization and a specialization process that recast the general form of each HCI design pattern into a concrete form. More precisely, the pattern instantiation component operates on two elements of a pattern template, including a name and a design solution element. The instantiated design patterns are thus generated as pattern fragments, from the present component, with specific knowledge regarding their name and solution.

As illustrated in Figure 7.2, the pattern instantiation procedure is achieved in three successive steps. The description of each step is provided below.

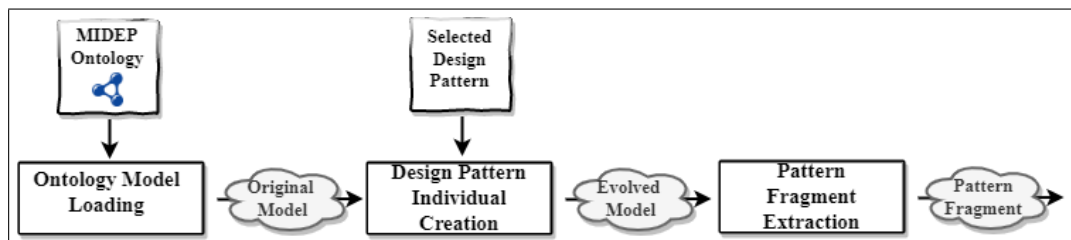


Figure 7.2: Pattern instantiation overview

- **Ontology model loading:** The objective of this step is to access the MIDEP OWL file and load the ontology model. This step is handled by using Jena API interfaces and methods to load the MIDEP ontology model.
- **Design pattern individual creation:** After loading the MIDEP ontology model, this step automatically creates new pattern instances that correspond to the selected HCI design pattern. The output of this step is the MIDEP ontology with the created design pattern instances. This step involves the definition of three statements : (i) a pattern instance for the "DesignPattern" class of the MIDEP ontology model, (ii) a pattern instance name, and (iii) a pattern instance solution.
- **Pattern fragment extraction:** After creating a pattern instance, this step is about extracting information regarding the created instance, including its name and design solution. To establish this step, a SPARQL query is executed through Jena against

the MIDEP ontology. Figure 7.3 outlines the SPARQL query considered in this step to extract pattern fragments.

```

PREFIX DP:<https://braam1.upv.edu.es/MIDEP/designPatternOnt.owl#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?name ?solution
  WHERE {
    ?designPatternInstance rdf:type DP:DesignPattern .
    ?designPatternInstance DP:DPName ?name .
    ?designPatternInstance DP:DPSolution ?solution .
  }

```

Figure 7.3: SPARQL query for extracting pattern fragments

7.2.2 Pattern integration

The second component within the ICGDEP system is responsible for the integration of design patterns into a UI model. This component takes as input the design pattern fragments, which are retrieved from the pattern instantiation component, and an initial UI model. Additionally, the pattern integration component maintains a repository of rules from which rules to manage pattern fragments are selected. As illustrated in Figure 7.4, the present component comprises two major phases, including "refine design pattern fragment" phase, and "assemble fragment into user interface model" phase. A detailed description of each phase is provided in the following subsections.

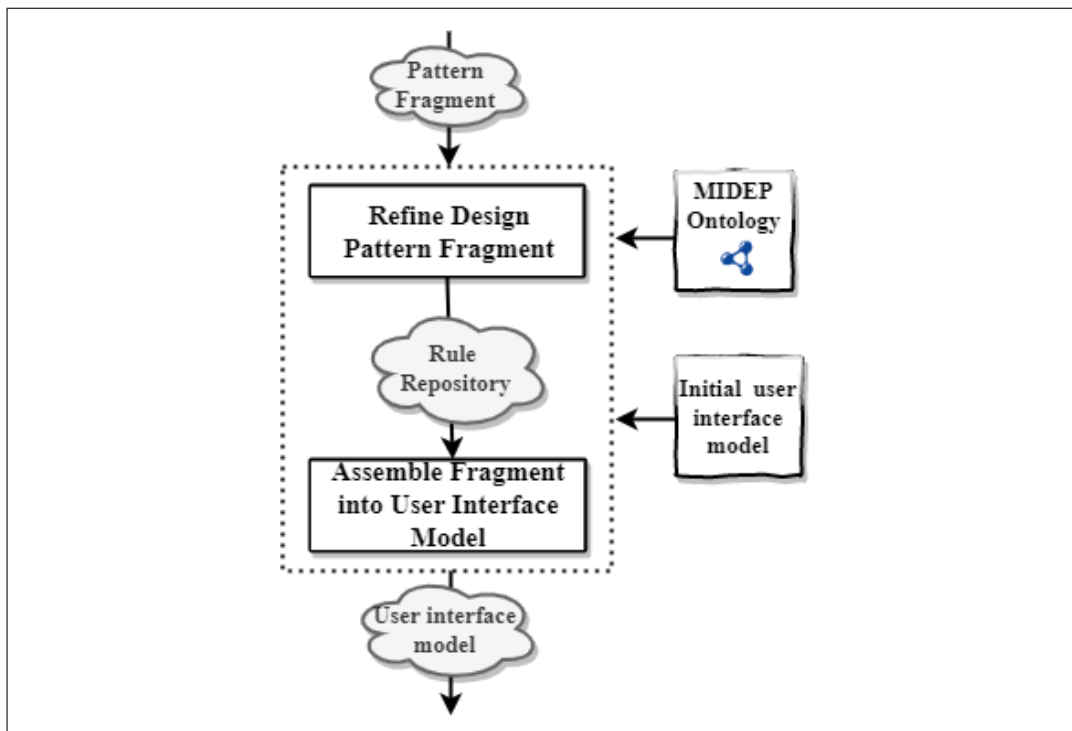


Figure 7.4: Pattern integration overview

7.2.2.1 Refine design pattern fragment

The first phase involved in the pattern integration component is about refining design pattern fragments. Particularly, it concerns the semantic parsing of design solutions, provided by each pattern fragments, through their transformations into inference rules. This phase focuses on building a repository of inference rules in which all possible design solutions that may be applied for integrating pattern fragments into a UI model are stored.

Figure 7.5, illustrates the proposed method for building rules from design solutions which requires the following four steps: (i) design solution preprocessing, (ii) semantic annotation, (iii) parsing into if-then statements, and finally (iv) transforming into inference rules. The details of the proposed method are presented as follows.

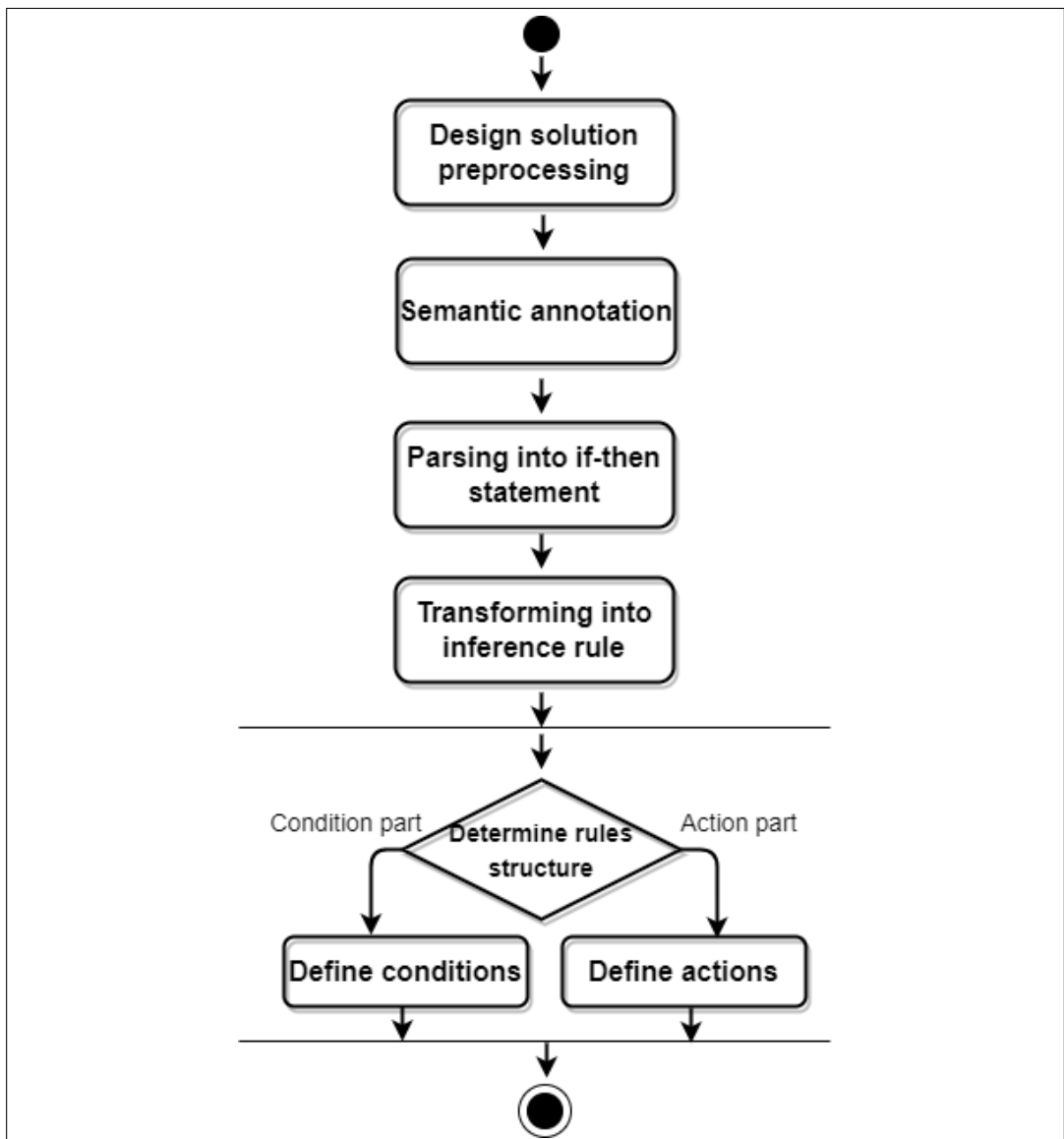


Figure 7.5: Workflow for refining pattern fragments

- **Design solution preprocessing**

Preprocessing design solutions constitutes the first step in refining design pattern fragments. It mainly attempts to accomplish the following three tasks: filter out irrelevant information or terms, analyze each design solution syntactically, and annotate the corpus of design solutions with a set of basic annotations that are further used in the semantic annotation phase. The standard GATE preprocessing resources are able to deal with these tasks. In particular, we make use of a range of ANNIE and Tool modules including Sentence Splitter, Tokenizer, Part Of Speech (POS) Tagger, and Morphological Analyzer. Figure 7.6 illustrates the flow used in the preprocessing step.

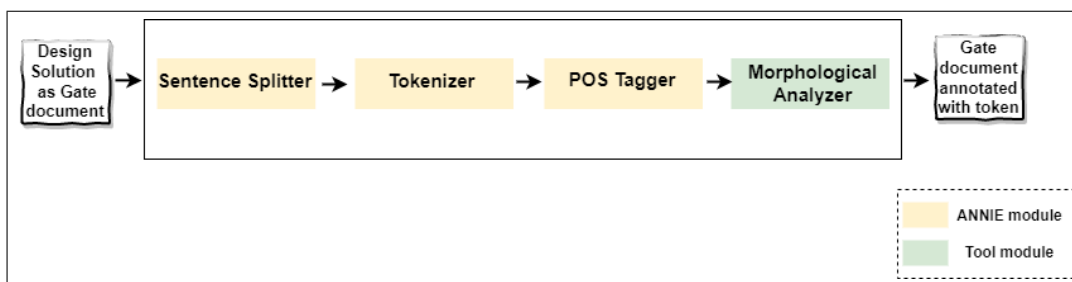


Figure 7.6: Design solution preprocessing major tasks

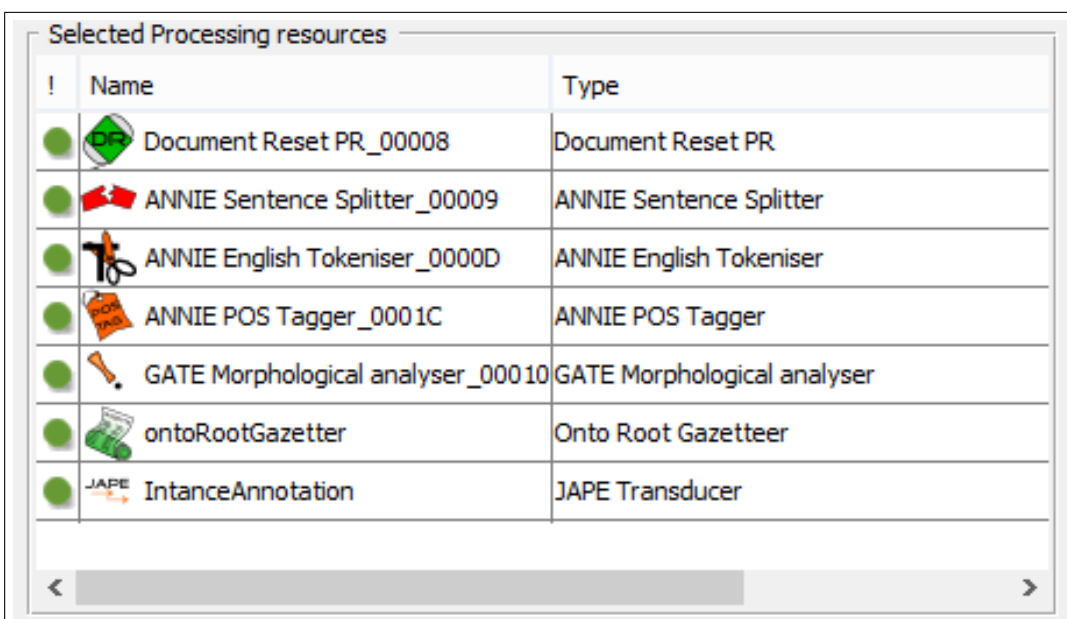
As can be seen in Figure 7.6, the process in GATE starts with the "Sentence Splitter", which creates sentence annotations, splits the corpus into sentences, and identifies each sentence in new line for the tokenizer. Then the "Tokenizer", which splits text into small textual fragments including symbols, words, phrases, or other meaningful elements called Tokens. It produces token and space token annotations with length, string, kind, and orthography features. Tokens can be numbers, words, symbols, or punctuation, while space tokens refers to white spaces. Then, tokens are processed by a "POS Tagger" that annotates tokens by assigning each token to its corresponding tag. GATE includes around 50 tags considering various parts of speech in order to generate the category feature to the annotated tokens. Some examples of POS tags are NN that identifies a noun, JJ that identifies an adjective, VB that refers to a verb at the base form, and PP that refers to a personal pronoun. Finally, the "Morphological Analyzer" is an integral part of the Tools plugin whose function is to analyze tokens annotated by POS tagger, specify the root base that corresponds to each word, and produce the root feature on token annotations.

- **Semantic annotation**

After preparing and processing design guidelines, the semantic annotation is intended to add semantics to text by identifying information from design solutions that can be associated with the MIDEF ontology entities presented in the design pattern fragments. This step can be defined as a type of information extraction that refers to the process of recognizing ontology concepts and instances in unstructured text. It aims to link entities and relations in the text to their semantic description in the ontology rather than just annotating words. As an information resource, this step requires the use of an ontology. The GATE tool is considered to perform the semantic annotation regarding the given ontology using the Onto-Root Gazetteer that allows performing ontology-based annotations. After loading

the MIDEP ontology as a language resource we make use of a Gate processing application for preprocessing the ontology resources and for creating ontology-based annotations regarding the given ontology. Figure 7.7 illustrates the GATE application pipeline for the semantic annotation step.

The Onto-Root Gazetteer annotates all features with their classifications using a set named "Lookup". Hence, the annotated features are divided based on their classifications. JAPE grammar is used to recognize and annotate the text of design solutions as the instances of the MIDEP ontology classes and instances. An-example of a JAPE rule for annotating ontology instances is shown in Figure 7.8.










!	Name	Type
	Document Reset PR_00008	Document Reset PR
	ANNIE Sentence Splitter_00009	ANNIE Sentence Splitter
	ANNIE English Tokeniser_0000D	ANNIE English Tokeniser
	ANNIE POS Tagger_0001C	ANNIE POS Tagger
	GATE Morphological analyser_00010	GATE Morphological analyser
	ontoRootGazetter	Onto Root Gazetteer
	IntanceAnnotation	JAPE Transducer

Figure 7.7: GATE application pipeline for semantic annotation

```
Phase: Instance_Phase
Input: Lookup
Options: control = appelt
Rule: InstanceLookup
({Lookup.type == "instance"}):label
-->
{
gate.AnnotationSet matchedAnns=
  (gate.AnnotationSet) bindings.get("label");
gate.Annotation matchedA =
  (gate.Annotation)matchedAnns.iterator().next();
gate.FeatureMap newFeatures= Factory.newFeatureMap();
outputAS.add(matchedAnns.firstNode(),
  matchedAnns.lastNode(), "Instance", newFeatures);
newFeatures.put("className",matchedA.getFeatures().get("classURI"));
}
```

Figure 7.8: Jape rule example for ontology instance annotation

- **Parsing into if-then statements**

The third step consists of parsing design solutions into if-then statements. It has the purpose of transforming the knowledge, identified in the semantic annotation step, into the IF-THEN rules. Consequently, this step enables the creation of several sets of rules for different design solutions. The designed rules are created following the suggestions given in the design solutions. The format of the resulting rules is composed with two parts, condition part and action part, in a form as follows: IF <condition(s)> THEN <action>. The condition part is matched to ontology knowledge, whereas the action part is a list of the design solutions. An example of inference rule (R1) in IF-THEN representation is given in Figure 7.9.

If patternInstance is a designPattern , interfaceElement is a UIElement
Then patternInstance isAssociatedTo interfaceElement

Figure 7.9: IF-THEN rule example

- **Transforming into inference rules**

In this step, the rules in the form IF-THEN are transported into inference rules. The aim behind is to build a rule repository that consists of several inference rules, which are defined in Apache Jena rule syntax. In this thesis, Jena is considered as a suitable rule engine since it has the ability to execute inference rules and to split the inferred knowledge from the base knowledge. The translation of IF-THEN rules into inference rules is achieved by a rule transformation algorithm so that they can be executed at run-time in the pattern integration component that uses the Jena inference engine. The rule transformation algorithm parses the condition and action parts of the IF-THEN rules and translates the condition variable and the action variable into Jena rule syntax. Figure 7.10 shows an inference rule example, named SampleRule1, where the above IF-THEN rule (R1) is transformed into an inference rule.

```
@include <OWLMicro>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix uni: <https://braam1.upv.edu.es/MIDEP/MIDEPontology.owl#>.
@prefix DP:<https://braam1.upv.edu.es/MIDEP/designPatternOnt.owl#>.
@prefix UI:<https://braam1.upv.edu.es/MIDEP/userProfileOnt.owl#>.

[SampleRule1:
  (?patternInstance rdf:type DP:DesignPattern)
  (?interfaceElement rdf:type UI:UIElement)
  ->
  (?patternInstance uni:isAssociatedTo ?interfaceElement)
]
```

Figure 7.10: Inference rule example

7.2.2.2 Assemble pattern fragment into UI model

The rule base built in the previous phase is executed in this phase to assemble pattern fragments into the UI model. Particularly, the inference rules aim to automatically associate the pattern fragments to UI elements using the “isAssociatedTo” object property. Additionally, a SPARQL query is formulated and communicated to the knowledge base to retrieve the required knowledge regarding the UI model. To accomplish this, Jena API is used, first, to load the MIDEP ontology owl file, then, to set the inference rules, and finally to execute the SPARQL query. Figure 7.11 highlights the three successive steps involved in the present phase.

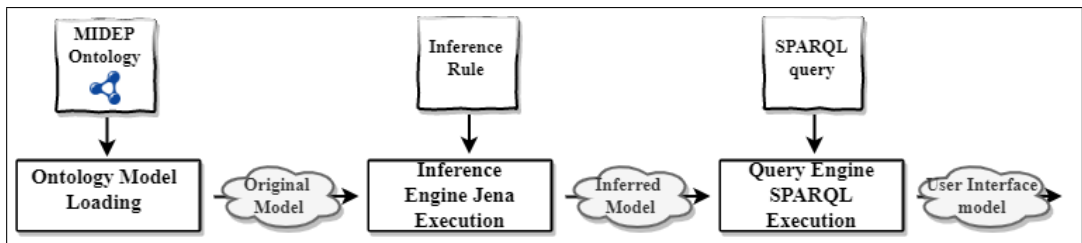


Figure 7.11: Assemble pattern fragment into UI model overview

7.2.3 User interface generation

The UI generation component represents the third component within the ICGDEP system architecture. Its main target is to use UI ontology models in combination with design patterns to generate UI source code automatically in the means of Ionic and Angular UI views. As described in the previous subsection, the UI models incorporate a set of HCI design patterns fragments and are modeled using ontologies expressed in OWL. The elements that constitute these UI models are transformed into Ionic and Angular UI views. To achieve this, the UI generation component is based on a generation method that relies on ontology-based and model-driven approaches. The fundamental idea behind adopting the ontology-based approach is the use of ontologies as the basis for constructing UI models. Whereas the model-driven approach is considered for code generation by ensuring the automatic transformations of models into source code. The process considered in the UI generation component is illustrated in Figure 7.12. The input to the present process is the UI model retrieved from the pattern integration component, whereas the resulting source code reflects Ionic and Angular UI views that can be automatically rendered on the target application using the global AUIDP framework. In the following subsections, a detailed description of the main phases responsible for the automatic UI generation, including UI model adaptation, model transformation, and source code generation phase, is provided.

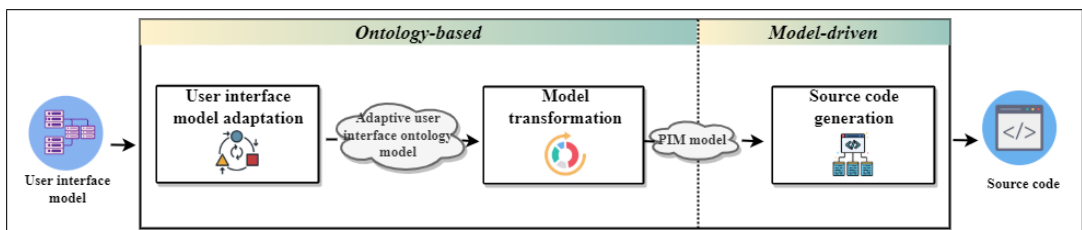


Figure 7.12: User interface generation main phases

7.2.3.1 User interface model adaptation

The aim of this phase is to ensure the adaptation of the current UI model by transforming the non-adapted UI ontology model to another one, defined as adaptive UI ontology model, according to the model retrieved from the pattern integration component. As illustrated in Figure 7.13, the adaptation process is achieved by two modules: A model checker and an adaptation engine.

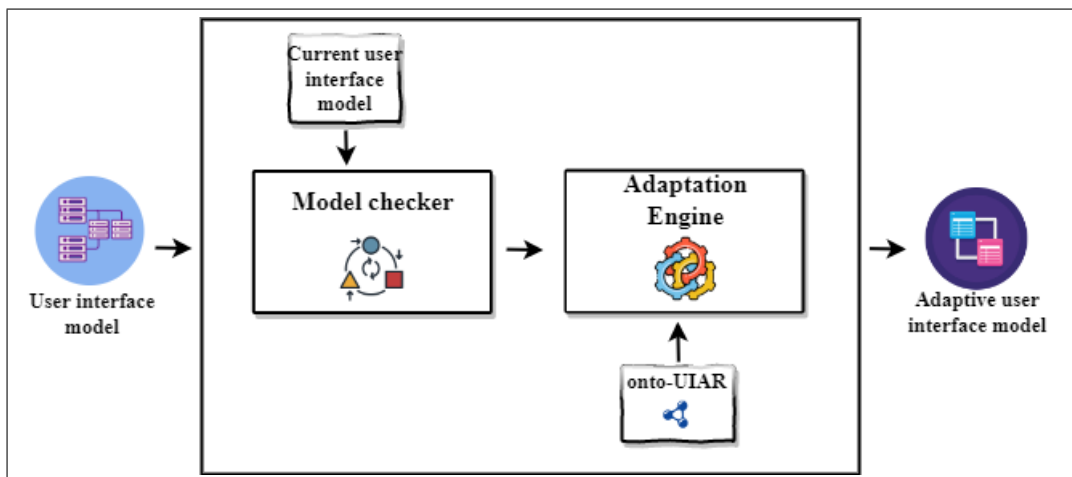


Figure 7.13: UI model adaptation overview

First, the model checker takes as input the UI ontology model, generated by the pattern integration component, and an instance of the current UI model to find the ontology elements that will be adapted. Then, the adaptation engine is in charge of performing the ontology model adaptations. Particularly, it uses an ontology named onto User Interface Adaption Rules (*onto-UIAR*), which represents concepts related to adaptation rules, adaptation actions, and adaptation services. This ontology is considered to retrieve the required adaptation services that enables the management of the current UI ontology model. The ontological model of the present *onto-UIAR* and the description of some ontology concepts are provided in Figure 7.14 and Table 7.1, respectively.

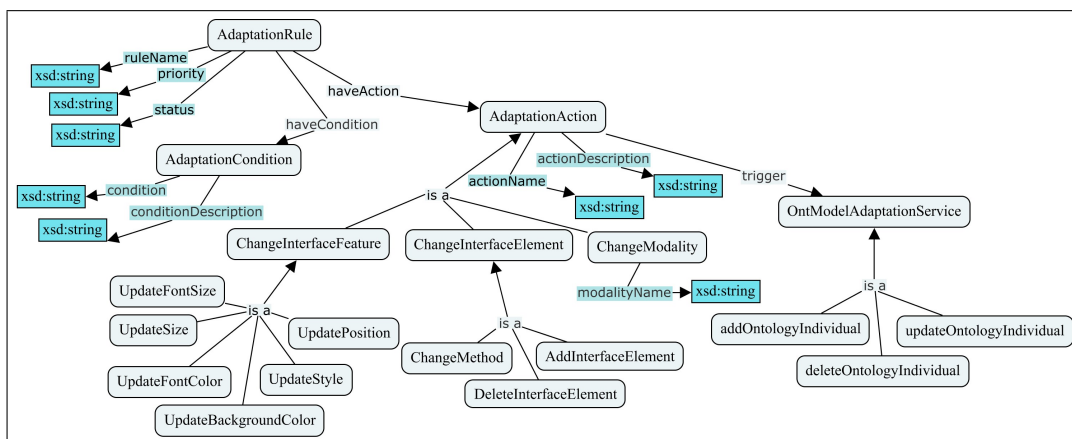


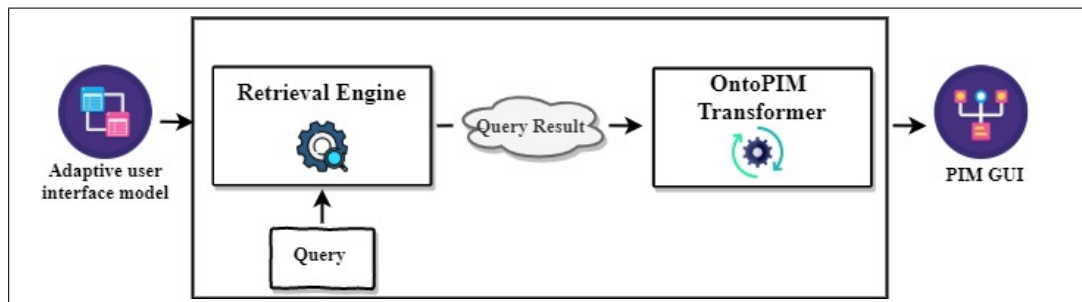
Figure 7.14: Onto-UIAR ontological model

Table 7.1: Onto-UIAR concept description

Concept	Description
AdaptationRule	Defines the adaptation rule that could be applied to the user interface ontology model
AdaptationCondition	A set of information that triggers adaptation rules
AdaptationAction	An adaptation action proposed by the adaptation rule, including ChangeInterfaceFeature, ChangeInterfaceElement, and ChangeModality
OntModelAdaptation-Service	Defines the ontology model adaptation services, used for the management of the user interface ontology model, including addOntologyIndividual, deleteOntologyIndividual, and updateOntologyIndividual

7.2.3.2 User interface model transformation

In this second phase, the UI ontology model, established from the previous phase, is transformed into a PIM GUI model, which is a platform independent model that includes elements to define the GUI. The model transformation phase is responsible to propagate the metadata about the UI from the ontology model and use it for generating a PIM, thus ensuring that the ontology knowledge provided during the first phase is present at the generated PIM GUI model. The goal of this second phase is to automatically build a PIM model for defining the UI. This phase is carried out using a model-driven transformation taking as input knowledge about the UI and producing the PIM GUI model as output. As shown in Figure 7.15, two main modules, including a retrieval engine, and an OntoPIM transformer, are considered in the present phase to perform the PIM GUI generation. A description of each module is provided below.

**Figure 7.15:** Model transformation overview

- The retrieval engine: This module is in charge of retrieving the ontology concepts required for the second module. To achieve this, Jena engine is used to execute SPARQL queries against the UI ontology model. The query results contain information regarding the UI ontology instances, data properties and object properties. Such information is given as input to the second module to build the PIM.
- The OntoPIM transformer: This module applies a set of transformation rules in order to perform the generation of the PIM GUI. Following rules, depicted in Figure 7.16, UI ontology model is transformed into PIM GUI. Particularly, all classes are mapped

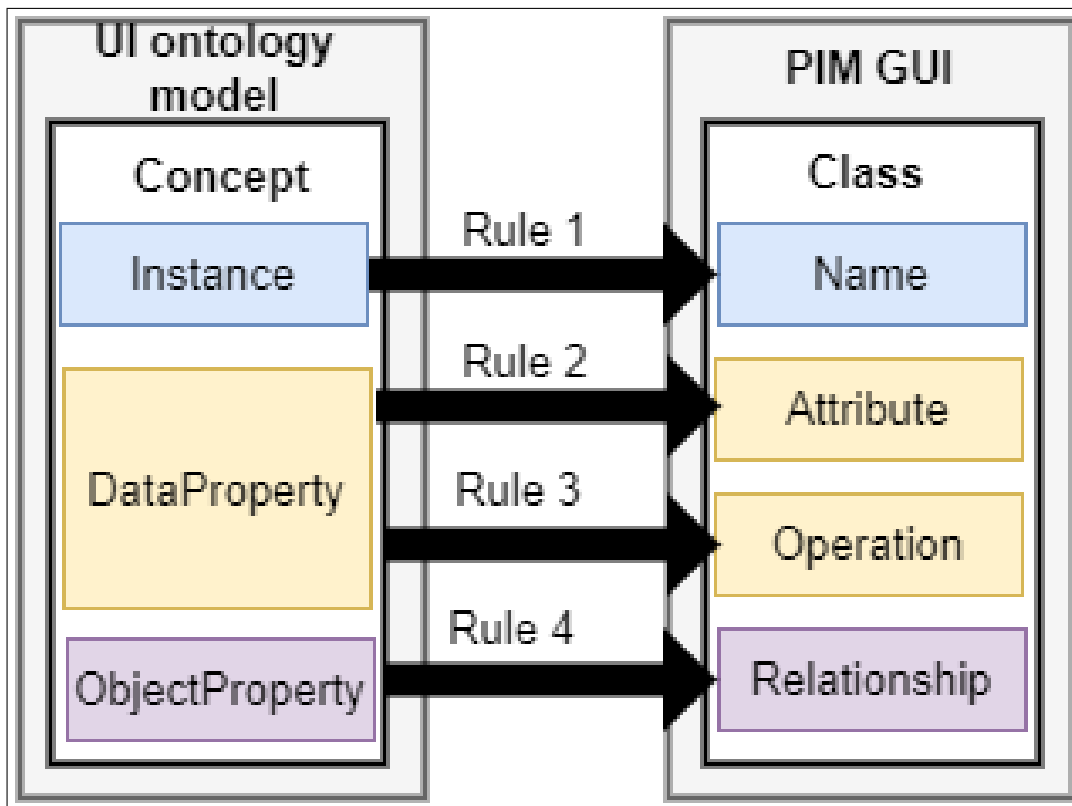


Figure 7.16: OntoPIM Transformation rules

from ontology's concepts under "Rule 1". In "Rule 2" and "Rule 3", attributes and operations come from ontology's data property. In addition, relationships can be generated under "Rule 4". The obtained PIM GUI model represents the specification of the GUI independently of the target platform and it is consistent with the DSL presented in the following phase.

7.2.3.3 Source code generation

On the basis of the results from the model transformation phase, the PIM GUI is used in this phase to allow the source code generation. The main goal of this phase is to automatically create the UI in the means of Ionic and Angular UI views. The UI view in Ionic and Angular applications are represented by two main parts, including a template and a component part. The template part contains the UI graphical view that includes UI display and interaction elements. Whereas, the component part defines the application logic. For addressing these aspects, this phase intends to build the structural aspects of Web and mobile applications and to generate its code automatically.

During the present source code generation phase, we propose a model-driven development approach based on a DSL in order to support code generation. Using this approach, the generation process can be structured in two levels: design-time and run-time level. This approach is depicted in Figure 7.17.

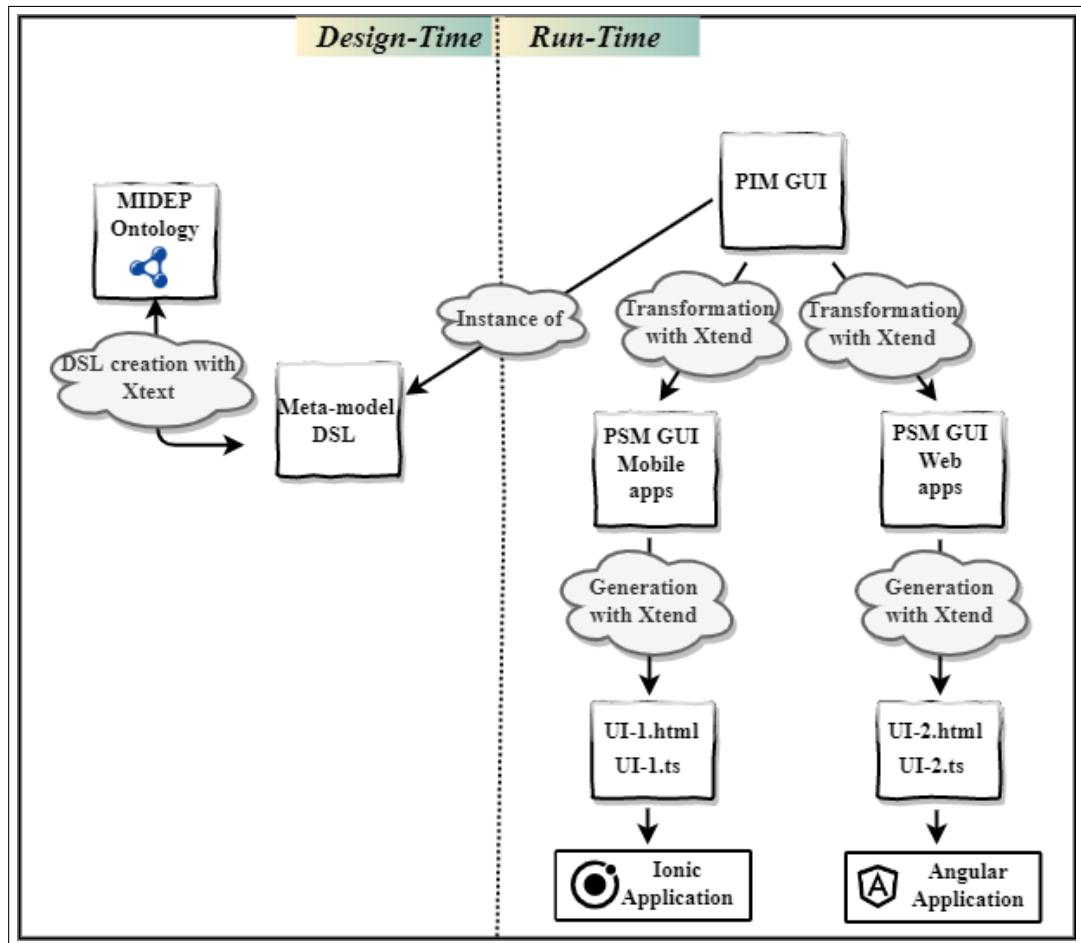


Figure 7.17: Source code generation overview

- **Level 1: Design-time**

At this level, a DSL to define the meta-model that describes the PIM GUI model is created using Xtext [Efftinge and Völter, 2006], which is a language development framework for creating DSLs. The grammar of the proposed DSL is defined using a GUI meta-model that is designed using some concepts of the MIDEP ontology. This meta-model is presented as a UML class diagram in Figure 7.18.

Each class in the GUI meta-model represents an Xtext rule. For example, the "Application" rule, depicted in Figure 7.19, is the starting Xtext rule of the proposed DSL. This rule contains one or more "ApplicationAttribute" rules that delegate to either the "ApplicationElementList" rule or "ApplicationMainPage" rule. Other Xtext rules are outlined in Figure 7.20. The "componentElement" rule describes a component element that can be button, spacer, text, etc. whereas; the "Button" rule represents an example of the component elements.

- **Level 2: Run-time**

At this level, the UI view for Ionic and Angular applications is automatically generated. This generation process is constituted of two principal steps:

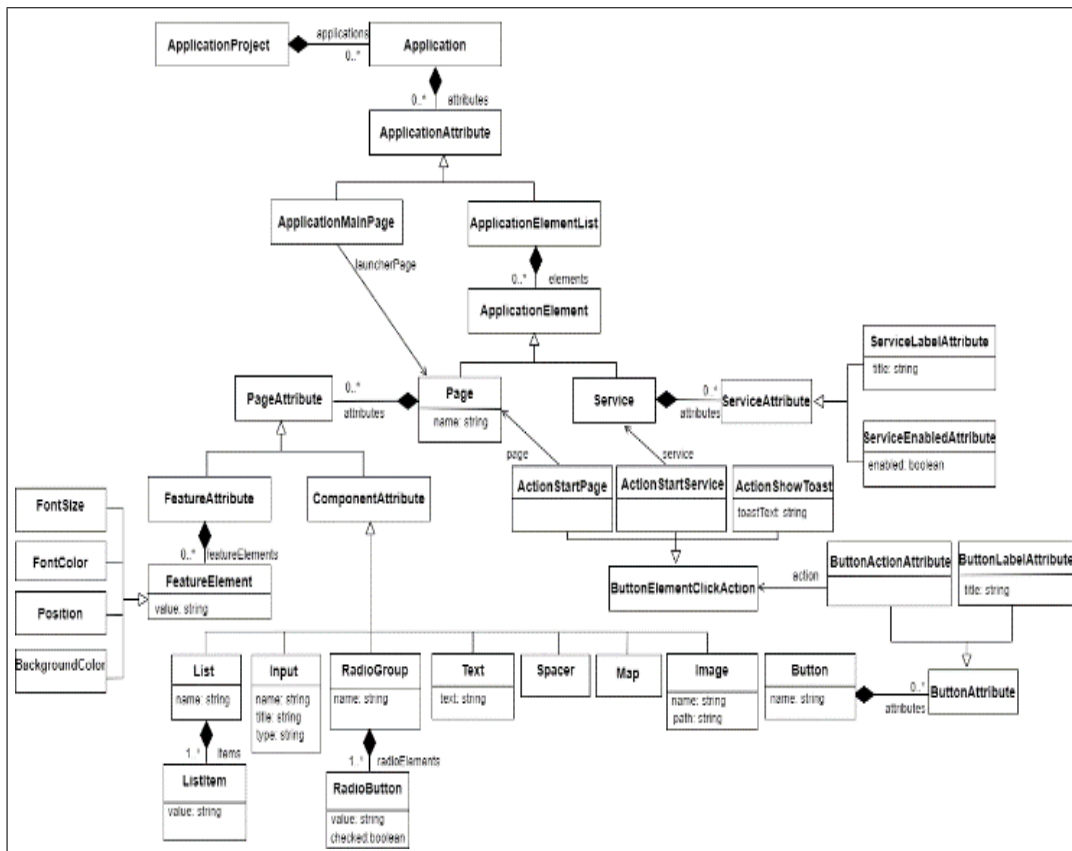


Figure 7.18: GUI meta-model

```

21 Application:
22   'application' name = PACKAGE_NAME
23   '{'
24     attributes += ApplicationAttribute (',' attributes += ApplicationAttribute)*
25   '}'
26 ;
27
28 ApplicationAttribute:
29   ApplicationElementList |
30   ApplicationMainPage
31 ;

```

Figure 7.19: MIDEPSDL grammar: Application rule

```

113 ComponentElement:
114   Button | Spacer | Text | Input | Image | RadioGroup | List | Map
115 ;
116
117 Button:
118   'button' name = ID
119   'size' size = STRING
120   '{'
121     attributes += ButtonAttribute (',' attributes += ButtonAttribute)*
122   '}'
123 ;

```

Figure 7.20: MIDEPSDL grammar: Component Element rule

- Transformation step: In this step, the PIM is transformed to Platform Specific Model (PSM) using the Xtend Language [Efftinge and Zarnekow, 2011]. Each element in the PIM is associated with a specific PSM element. Table 7.2 presents some transformation rules based on the PIM toward the targeted models.
- Projection step: Once PIM to PSM transformations are made, the projection step is performed to generate the source code from the PSM. This step is carried out using transformation templates created by Xtend. An extract of the transformation templates written using Xtend expressions is shown in Figure 7.21.

Table 7.2: Some transformation rules for Web (Angular) and mobile (Ionic) applications

PIM	PSM GUI for Angular	PSM GUI for Ionic
Page	Page	Page
Text	Label	Ion-label
Input	Input	Ion-input
Type = text	Type = text	Type = text
Type = password	Type = password	Type = password
Type = email	Type = email	Type = email
Type = number	Type = number	Type = number
Button	Button	Button ion-button
Spacer	Br	Br

```

filesystem.generateFile(String.format("%s/src/angularPage/%s.html", projectName,
    javaToAndroidIdentifier(page.name)
),
    generateAngularLayoutComponent(page)
);

filesystem.generateFile(String.format("%s/src/ionicPage/%s.html", projectName,
    javaToIonicIdentifier(page.name)
),
    generateIonicPageLayoutComponent(page)
);

```

Figure 7.21: Extract of the transformation template

7.3 ICGDEP Implementation

The proposed ICGDEP system has been implemented into a tool called ICGDEPTool to generate UI source code from the selected HCI design patterns in a fully automatic fashion.

7.3.1 ICGDEP tool

The developed tool provides three main functionalities. First, pattern instantiation is provided to give pattern fragments to users. This functionality considers the phase of instantiating design patterns, which are selected by the IDEPAR system presented in Chapter 6. After exiting this phase, users will get pattern fragments ready for the next

phase. Second, the pattern integration functionality is considered to provide an instance of the UI ontology model. It does so by implementing the pattern integration process, which responds by creating the UI ontology model ready for source code generation. Finally, the third functionality is responsible for generating the UI views for the target platforms. This functionality communicates with the UI generation process to get the applications' source files. All functionalities are developed in Java applications and exported as JAR files. These Jar files are uploaded into a server Application developed using JAVA. In this case, the server application contains three JAR files, namely PatternInstantiation.jar, PatternIntegration.jar, UserInterfaceGeneration.jar, and commands to start the ICGDEP system. The server application launches the ICGDEP system using the JAVA API in a separate JVM. Figure 7.22 depicts the ICGDEPTool architecture with its main components. The main role of this tool is to send input of users to the ICGDEP system in order to get the generated UI source code.

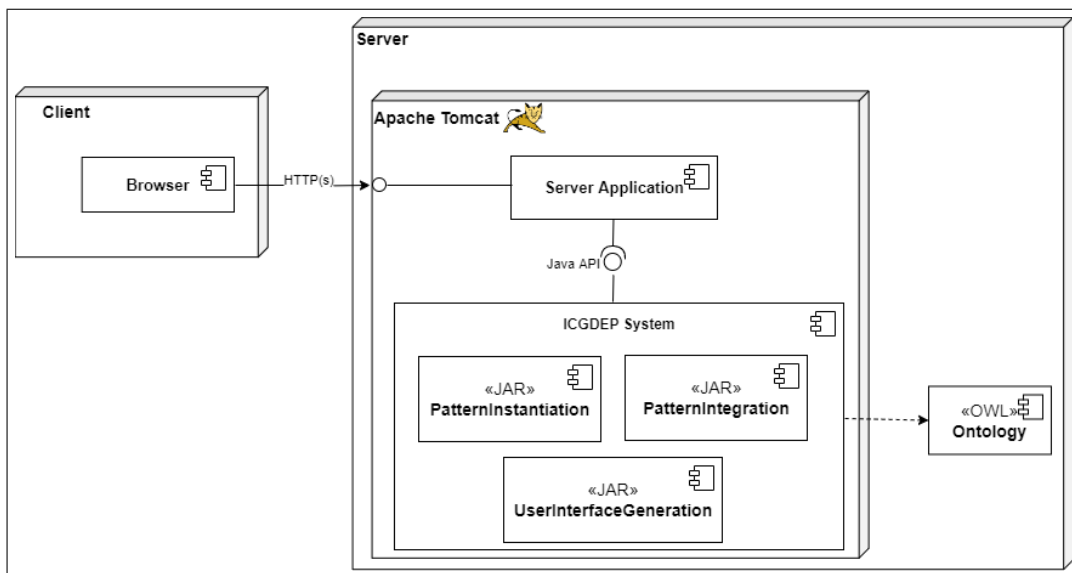


Figure 7.22: ICGDEPTool architecture

7.3.2 User interface generation example

In this subsection, we present two examples in which the ICGDEPTool is used by a developer to obtain the UI source code from design patterns conforming to a specific context situation. The aim of this case study is to highlight how the developed ICGDEP tool can automatically generate the UI views of an Ionic application.

In the first case, the developer wants to get the UI for solving the design problem DPS-2, described in Chapter 6. First, using the IDEPAR application, presented in the previous chapter, the developer gets the relevant design patterns for DPS-1. Then, the developer interacts with the ICGDEP tool and enters the recommended design patterns, including ZoomInDP, ZoomOutDP, MapNavigatorDP, DarkBackgroundColorDP, and WhiteColorDP. The process achieved by the ICGDEP tool that allows UI views generation for the design patterns entered by the developer, is illustrated in Figures 7.23, 7.24, and 7.25.

As shown in Figure 7.23, the design patterns are pushed to the pattern instantiation component (Figure 7.23), which provides an instance of pattern fragments. Then, these

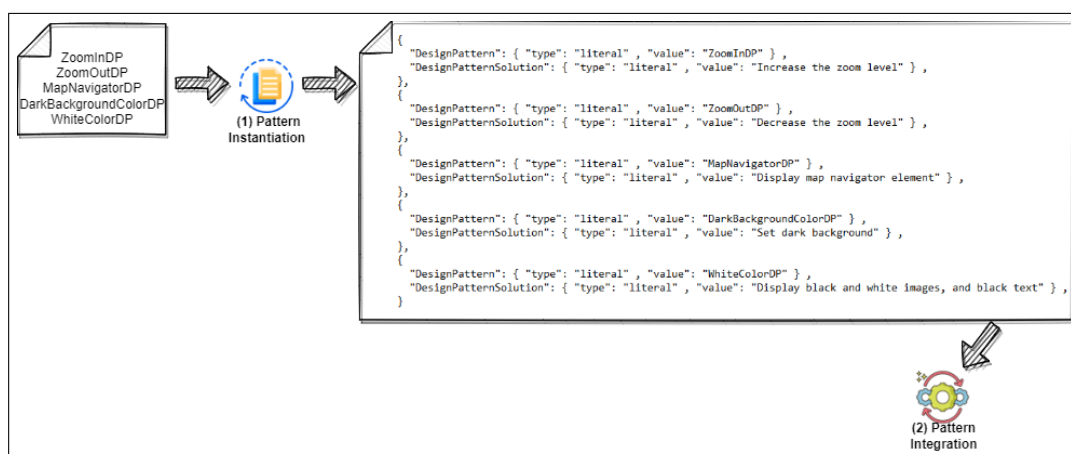


Figure 7.23: UI generation example for DPS-1: Results of the pattern instantiation component

fragments are transferred to the integration component (Figure 7.24), to provide a UI ontology model that is used in the UI generation component (Figure 7.25). There, the UI ontology model is adapted to an adaptive UI model (Figure 7.25-i) and transformed to a PIM GUI model (Figure 7.25-ii). Finally, the UI code of Ionic application is generated following the source code generation phase that allows creating the source code based on the obtained PIM GUI model (Figure 7.25-iii). The screenshot of the resulting UI, within the Ionic application, is depicted in Figure 7.25-iv.

In the second case, the developer uses the ICGDEP tool in order to get the UIs that solve the design problem DPS-2, described in Chapter 6. First, using the IDEPAR application, presented in the previous chapter, the developer gets the relevant design patterns for DPS-2. Then, the developer interacts with the ICGDEP tool and enters two design patterns, including LoginDP and AudioInputDP. Figure 7.26 presents the process achieved by the ICGDEP tool that enables the generation of UI views that corresponds to the design problem DPS-2.

7.4 Concluding Remarks

In this chapter, a generation system for creating UI views for mobile and Web applications has been introduced. The present system, named ICGDEP, includes three main components: (i) the pattern instantiation component, where the design patterns from the IDEPAR system and their corresponding design solutions are extracted, (ii) the pattern integration component, where the design pattern fragments are integrated and represented in a set of UI ontology models, (iii) and the UI generation component in which the code that implements the target mobile and Web applications is generated.

At the end of this chapter, we have described a tool, named ICGDEPTool, which provides support to work with the ICGDEP system to enable the generation of different UI views. The use of the developed tool is illustrated with an example in which a UI view for an Ionic application is created by our tool that is able to perform code generations from the name of design patterns given by the developer.

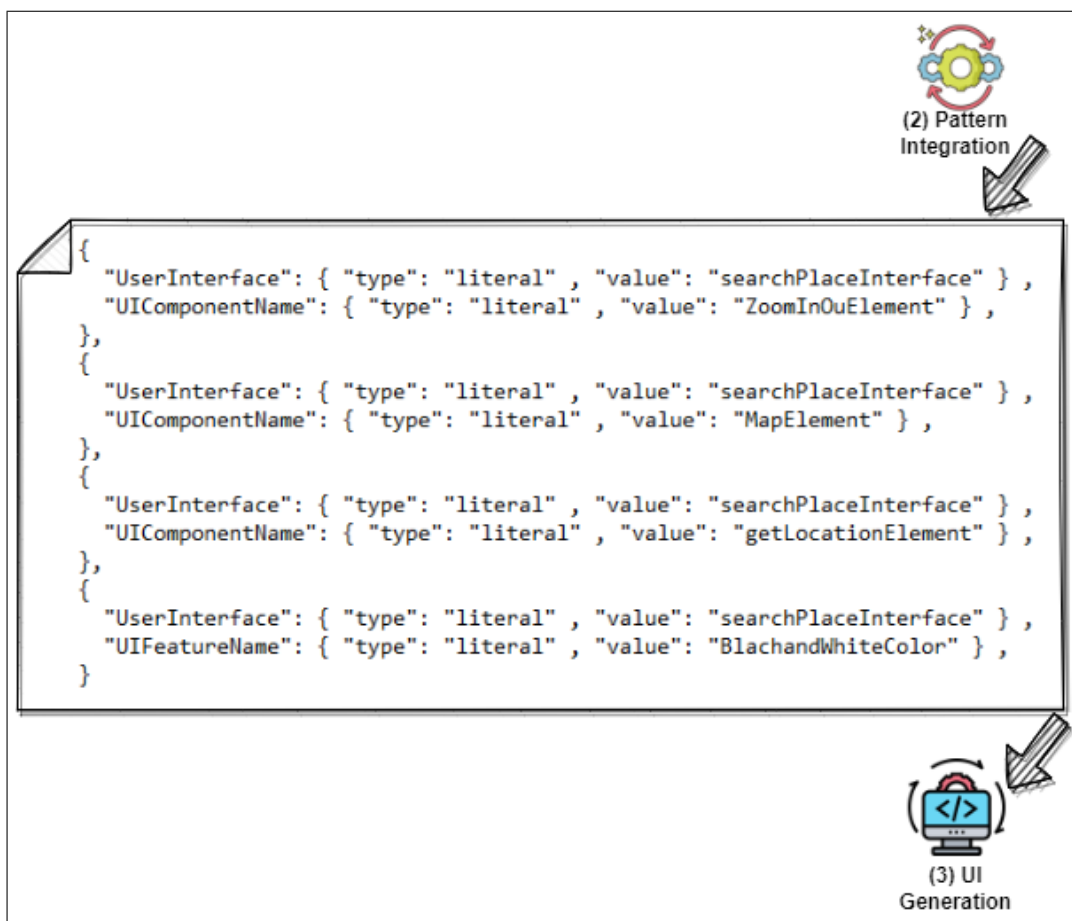


Figure 7.24: UI generation example for DPS-1: Results of the pattern integration component

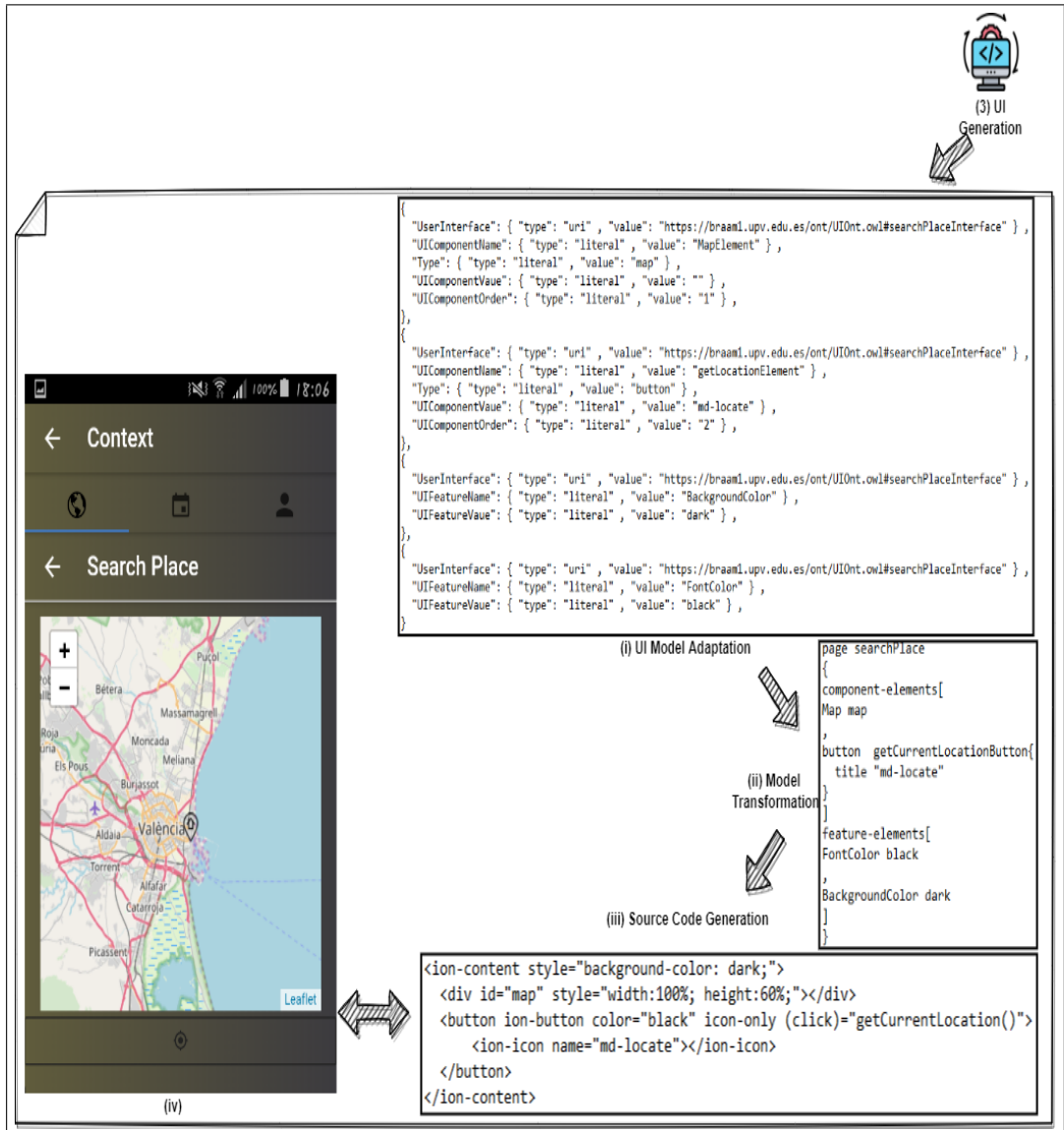


Figure 7.25: UI generation example for DPS-1: Results of the UI generation component

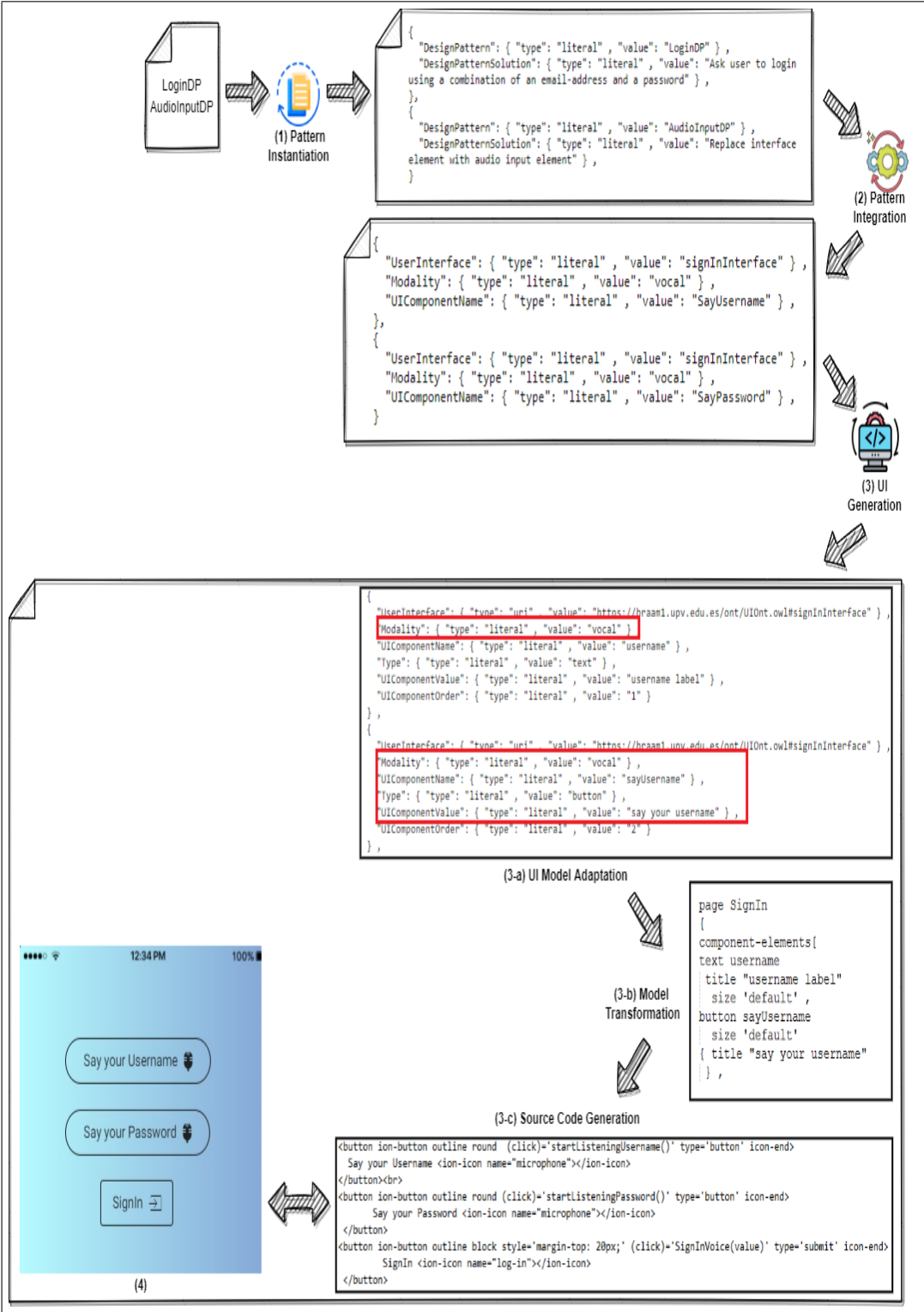


Figure 7.26: UI generation example for DPS-2

Evaluation

8.1 Introduction

This chapter deals with the evaluation performed to validate the work presented in this dissertation thesis. Specifically, this work has been validated under different perspectives according to the confronted research questions defined in Chapter 1.

Firstly, in Section 8.2, we present our evaluation for the MIDEP ontology. Then, in Section 8.3, we report on expert-based and user-centric evaluation studies for the IDEPAR system. Next, In Section 8.4, the evaluation of the ICGDEP system is demonstrated through a scenario example. Furthermore, in Section 8.5, we present our usability and user satisfaction evaluations for the applications generated by the global AUIDP framework. Finally, Section 8.6 concludes this chapter by discussing and summarizing the obtained findings of the present evaluation.

8.2 Evaluation of the MIDEP Ontology

8.2.1 Evaluation overview

The evaluation of the MIDEP ontology comprehends the ontology verification, validation, and assessment activities [Suárez-Figueroa et al., 2012]. Ontology verification confirms that the ontology is built correctly, whereas ontology validation and ontology assessment aims to prove that the ontology really models the world for which the ontology was created and to check the ontology against users' requirements, respectively. Considering these activities, the purpose of the present section is to evaluate the MIDEP ontology in order to answer the research questions RQ-1-2 and RQ-1-3 introduced in Chapter 1. In particular, we focus (i) on verifying whether the developed ontology is correct according to three evaluation criteria, namely completeness, conciseness, and consistency, and (ii) on checking how effective the ontology is in the context of different applications.

In this regard, the MIDEP ontology is evaluated by using three approaches: competency questions, technology-based, and application-based evaluations. First, the ontology is evaluated against competency questions and thereby attempts to establish whether the ontology could answer the defined competency questions to ascertain the completeness of

the ontology. Afterward, a technology-based evaluation approach is addressed with respect to specific ontology quality criteria in order to ensure that the ontology is rid from pitfalls. At last, the ontology is evaluated using an application-based approach to assess the ability of the ontology to serve as a knowledge base for a computer system. The present section concludes with a discussion.

8.2.2 Competency questions evaluation

In the present evaluation, competency questions are proposed to evaluate the ontology since they play a major role in the process of ontology development [Grüninger and Fox, 1995]. The ability to answer a competency question meaningfully can be regarded as a requirement that an ontology must satisfy. Thus, this study focuses on reformulating competency questions as queries to retrieve data from the ontology and to verify whether the competency questions are answered or not. In this sense, queries are written in SPARQL language, which is a semantic query language used for interrogating ontologies. This process requires an ontology editor in order to run queries, specifically the Protégé tool is used for formulating queries and for visualizing results.

To evaluate the capability of the MIDEF ontology to answer the competency questions derived from the ORSD, presented in Chapter 5, each question is converted into SPARQL queries and implemented in Protégé using the SPARQL Query plugin. In the following, five examples of competency questions, which cover the design patterns module, are provided along with their corresponding SPARQL queries and results.

- **CQ1:** What are the design patterns contained in the ontology? Figure 8.1 presents the SPARQL query formalizing competency question CQ1 to retrieve all instances of the DesignPattern concept. The result of this query, as illustrated in the figure, contains the 45 HCI design patterns modeled in the MIDEF ontology.
- **CQ2:** What are the concepts modeled in the ontology that can be used to categorize design patterns? Figure 8.2 displays the formal representation of this competency question using SPARQL query. The query asks for the subclasses of the class DesignPattern that are linked to the DesignPatternsGroup class via the object property hasDPgroup. The result of this query, as shown in the figure, contains the category and the group for each design pattern modeled in the MIDEF ontology.
- **CQ3:** What are the concepts represented in the ontology that model a design problem? The SPARQL query that corresponds to this competency question CQ3 is presented in Figure 8.3. The query asks for the subclasses of the class ProblemConcept. Consequently, the result of this query as shown in the figure contains all instances of the ProblemConcept.
- **CQ4:** What are the design patterns that provide a solution to the problem of colorblindness? Figure 8.4 shows the SPARQL query formalizing

SPARQL query:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX DP: <https://braam1.upv.edu.es/MIDEP/designPatternOnt.owl#>

SELECT ?designPattern
      WHERE { ?designPattern rdf:type DP:DesignPattern }

```

designPattern
DP:MapNavigatorDP
DP:BookingDP
DP:NavigationTabDP
DP:AudioInputDP
DP:SlidesShowDP

Figure 8.1: SPARQL query results for CQ1

SPARQL query:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX DP: <https://braam1.upv.edu.es/MIDEP/designPatternOnt.owl#>
SELECT ?designPattern ?category ?group
      WHERE { ?category rdfs:subClassOf DP:DesignPattern .
              ?designPattern rdf:type ?category .
              ?group rdf:type DP:DesignPatternGroup .
              ?designPattern DP:hasDPgroup ?group
            }

```

designPattern	category	group
DP:GalleryDP	DP:InteractionDP	DP:BasicInteraction
DP:SearchBoxDP	DP:InteractionDP	DP:Search
DP:DarkColorDP	DP:CustomizationDP	DP:FontColor
DP:ProcessingPageDP	DP:InteractionDP	DP:Feedback
DP:FontSizeMediumDP	DP:CustomizationDP	DP:FontSize
DP:RatingDP	DP:InteractionDP	DP:MakingChoice
DP:EmptyStateDP	DP:InteractionDP	DP:EmptyState
DP:NotificationDP	DP:InteractionDP	DP:Notification

Figure 8.2: SPARQL query results for CQ2

The screenshot shows a SPARQL query window with the following content:

```
SPARQL query:
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX DP: <https://braam1.upv.edu.es/MIDEP/designPatternOnt.owl#>
SELECT ?problemConcept
  WHERE {
    ?problemConcept rdfs:subClassOf DP:ProblemConcept .
  }
```

The results table below the query is as follows:

problemConcept
DP:ApplicationPurpose
DP:SrcCodeConstraint
DP:UserCharacteristic
DP:UserInterfaceIssue
DP:ApplicationFunctionality

Figure 8.3: SPARQL query results for CQ3

this competency question CQ4. The query asks for subclasses of DesignPattern class, which are linked to DesignProblem class with the object property `isSolutionTo`. Additionally, it asks only design patterns related with the colorblindness concept that are linked via the data property `problemDescription`. As illustrated in Figure 8.4, executing the present query on the MIDEP ontology returns `DarkBackgroundColorDP` and `BlackandWhiteColorDP` design patterns.

The screenshot shows a SPARQL query window with the following content:

```
SPARQL query:
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX DP: <https://braam1.upv.edu.es/MIDEP/designPatternOnt.owl#>
SELECT ?designPattern ?solution
  WHERE {
    ?designPattern rdf:type DP:DesignPattern .
    ?problem rdf:type DP:DesignPatternProblem .
    ?problem DP:DPproblemDescription ?problemDescription .
    ?designPattern DP:isSolutionTo ?problem .
    ?designPattern DP:DPSolution ?solution .
    FILTER (?problemDescription='colorblindness')
  }
```

The results table below the query is as follows:

designPattern	solution
DP:DarkBackgroundColorDP	"Set dark background"^^<http://www.w3.org/20
DP:BlackandWhiteColorDP	"Display black and white images, and black text"^^

Figure 8.4: SPARQL query results for CQ4

- **CQ5:** What are the design patterns that provide a solution to the problem of low vision? The SPARQL query formalizing this competency question CQ5 is shown in Figure 8.5. The query asks for subclasses of DesignPattern class, which are linked to DesignProblem class with the object property `isSolutionTo`. Moreover, the present SPARQL query asks only design patterns that are related with the low vision concept linked via the data property `problemDescription`. Executing this present query on the MIDEP ontology returns `FontSizeLargeDP` and `ZoomInDP` design patterns.

designPattern	solution
DP:FontSizeLargeDP	"Maximize Text font size and interaction element s
DP:ZoomInDP	"Increase the zoom level"^^<http://www.w3.org

Figure 8.5: SPARQL query results for CQ5

All queries for each competency question were executed in Protégé and the corresponding results were obtained. Figures 8.1, 8.2, 8.3, 8.4, and 8.5 show screenshots of SPARQL queries and output results for competency questions CQ1-CQ5 on running the queries in Protégé. All queries are successfully executed over the developed ontology. From the SPARQL query results, it can be seen that the MIDEP ontology is able to answer competency questions by generating data. Through this evaluation, the obtained results demonstrate the capability and the proficiency of the MIDEP ontology for answering the competency questions. These results also indicate that the consistency between the retrieved data and the ontology concepts proves the ability of the developed ontology to provide correct answers for each competency question. Moreover, the present study reveals that the defined SPARQL queries can suitably interrogate the MIDEP ontology and retrieve the correct data. Consequently, the data provided by the ontology to answer the competency questions verifies the completeness of the MIDEP ontology.

The present competency question evaluation approach shows how queries could be expressed with SPARQL to retrieve data from the developed ontology and to answer competency questions. In particular, the evaluation by executing SPARQL queries satisfied the results for the functionality of the MIDEP ontology. This reveals that knowledge represented in the ontology is sufficient to answer SPARQL queries translated from competency questions. As a result of the present evaluation, the MIDEP ontology is able to solve competency questions completely. This implies that the developed ontology meets the completeness standards.

8.2.3 Technology-based evaluation

The present evaluation aims to ensure that the developed ontology is free from the critical pitfalls in order to verify its consistency, completeness, and conciseness. Different tools have been developed to support the evaluation of ontologies using the technology-based approach. In this study, the MIDEP ontology is evaluated through the OOPS! tool, which is a web-based evaluation tool used for the detection and the identification of common pitfalls and modeling issues in OWL ontologies. This tool is applied since it allows the automatic detection of greater number of common pitfalls in ontologies compared with other tools [Poveda-Villalón et al., 2014].

The screenshot shows the OOPS! Ontology Pitfall Scanner interface. At the top, it says "OOPS! (Ontology Pitfall Scanner!) helps you to detect some of the most common pitfalls appearing when developing ontologies." Below this, there are input fields for "Scanner by URI" and "Scanner by RDF". The "Scanner by URI" field contains the example URI: `http://oops.linkeddata.es/example/swc_2009-05-09.rdf`. The "Scanner by RDF" field contains an XML snippet:

```
<?xml version="1.0"?>
<rdf:RDF xmlns="https://braam1.upv.edu.es/MIDEP/MIDEPOntology.owl#"
  xml:base="https://braam1.upv.edu.es/MIDEP/MIDEPOntology.owl"
  xmlns:UI="https://braam1.upv.edu.es/MIDEP/userInterfaceOnt.owl#"
  xmlns:rdfs="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xm1="http://www.w3.org/XML/1998/namespace"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:UP="https://braam1.upv.edu.es/MIDEP/userProfileOnt.owl#"
  >
```

Below the input fields, there is a checkbox labeled "Unchecked this checkbox if you don't want us to keep a copy of your ontology." and a link "Go to advanced evaluation".

The "Evaluation results" section shows a summary of pitfalls:

- Critical** (red circle): It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- Important** (orange circle): Though not critical for ontology function, it is important to correct this type of pitfall.
- Minor** (yellow circle): It is not really a problem, but by correcting it we will make the ontology nicer.

The results table shows:

Results for P08: Missing annotations.	183 cases Minor
Results for P10: Missing disjointness.	ontology* Important
Results for P21: Using a miscellaneous class.	1 case Minor
Results for P41: No license declared.	ontology* Important

On the right side, there is a "Want to help?" section with links for "Suggest new pitfalls" and "Provide feedback". Below that is a "Documentation:" section with links for "Pitfall catalogue", "User guide", and "Technical report". At the bottom right, there is a "Related papers:" section.

Figure 8.6: OOPS! evaluation summary before corrections

After executing the OOPS! tool with the MIDEP ontology, a summary of the pitfalls encountered is generated as can be seen in Figure 8.6. Specifically, OOPS! displays the results for each pitfall in three ways according to the kind of pitfall, namely critical, important, and minor. As illustrated in Figure 8.6, the generated summary attains zero critical pitfalls, two minor pitfalls and two important pitfalls. The two minor pitfalls, each resulted from missing annotations (P08) and using a miscellaneous class (P21). Whereas the two important pitfalls are identified with respect to missing disjointness (P10) and no license declared (P41).

The pitfalls detected by OOPS! can also be classified based on the following evaluation criteria: consistency, completeness, and conciseness. The obtained results show that no consistency nor conciseness pitfalls are detected. Nevertheless, other pitfalls are detected (P08, P10, P21, P41) and one of them (P10) is related to the ontology completeness. Table 8.1 presents a rundown of the four pitfalls encountered.

Given the report of the OOPS! tool, the required modifications to the MIDEP ontology are made in order to fix the detected pitfalls, including P08, P10, P21, and P41. In the following, a description of each of these four pitfalls along with the provided solutions are presented.

- **P08:** This minor pitfall is about missing annotations. It indicates that the ontology elements lack annotation properties that label or define them. In order to address this issue, the annotation properties of “`rdfs:label`” and “`rdfs:comment`” were considered to define annotations of the MIDEP ontology elements.
- **P10:** This important pitfall is about missing disjointness. It reveals that the ontology lacks disjoint axioms between classes or properties. To fix this pitfall, disjoint axioms

Table 8.1: MIDEP ontology pitfalls detected by OOPS!

Criteria	Pitfall Description	Level	Appears In
Consistency	No detected pitfalls that correspond to consistency	-	0 cases
Completeness	P10: Missing disjointness	minor	This pitfall applies to the ontology in general
Conciseness	No detected pitfalls that correspond to conciseness	-	0 cases
Other Pitfalls	P08: Missing annotations	minor	183 cases
	P21: Using a miscellaneous class	minor	1 case
	P41: No license declared	important	This pitfall applies to the ontology in general

between `AdaptationStrategy` and `SelectionStrategy` classes were added.

- **P21:** This minor pitfall is about using a miscellaneous class. It indicates that the ontology includes a class named `Other` or `Miscellaneous`. To fix this pitfall, the class `MiscellaneousDP` was renamed to `GenericDP`.
- **P41:** This important pitfall is about no license declared. It reveals that the license declaration is missing from the ontology metadata. To address this pitfall the license information was added in the MIDEP ontology.

After making the required modifications in Protégé, OOPS! suggests the reexamination of the MIDEP ontology in order to verify whether the modifications were executed correctly. Figure 8.7 illustrates that the MIDEP ontology is able to repair all pitfalls, including P08, P10, P21, and P41. Consequently, the MIDEP ontology is free of pitfalls and meets all the evaluation criteria of the OOPS!.

Results of the present evaluation using the OOPS! tools reveal that the MIDEP ontology is free from all pitfalls, complete and meets the conciseness and consistency standards, which preserve the correctness of the developed ontology.

8.2.4 Application-based evaluation

In the present evaluation approach, the MIDEP ontology is validated by providing the following applications and systems.

- **MIDEP database and interface:** A Web-based application, which enables the visualization of knowledge modeled in the MIDEP ontology, was developed. The interfaces provided by the application are designed to expose the categories and groups for each design pattern instance. Such interfaces display design pattern instances with their corresponding descriptions, including their name, problem, and solution. Figure 8.8 and Figure 8.9 show interface examples of the Web application: interface for design pattern groups that belong to interaction category (Figure 8.8) and interface for information regarding the design pattern named audio input (Figure 8.9). Additionally, RESTful Web services were developed to create, retrieve, delete,

The screenshot displays the 'Ontology Pitfall Scanner!' web application. At the top, it explains that OOPS! helps detect common pitfalls in ontologies. Below this, there are two input methods: 'Scanner by URI' and 'Scanner by RDF'. The 'Scanner by RDF' method is selected, and a text area contains an RDF/XML snippet with various namespace declarations. A 'Scanner by RDF' button is visible to the right of the text area. Below the text area, there is a checkbox labeled 'Uncheck this checkbox if you don't want us to keep a copy of your ontology.' and a link 'Go to advanced evaluation'. The main section is titled 'Evaluation results' and contains a 'Congratulations!' message stating that the ontology does not contain any bad practice detectable by OOPS!. It also provides information about how OOPS! identifies pitfalls and a badge for 'PITFALL FREE' ontologies. On the right side, there are sections for 'Want to help?' (with links for 'Suggest new pitfalls' and 'Provide feedback'), 'Documentation:' (with links for 'Pitfall catalogue', 'User guide', and 'Technical report'), and 'Related papers:'.

Figure 8.7: OOPS! evaluation summary after corrections

update and query the MIDEF ontology metadata. Finally, a Java Gate application is developed with focus on populating the MIDEF ontology with instances from existing design pattern catalogs. The application offers the possibility of populating the ontology with the catalog whose path is entered by the user. The process of populating the ontology is achieved using various functionalities of GATE API, e.g. ANNIE, Ontology_Tools, Gazetteer_Ontology_Based, and Gazetteer_LKB.

In order to illustrate the ontology population with a specific design pattern that is available from accessible Web sites, we tested our application with "LiveFilter" patterns using the following link: <https://ui-patterns.com/patterns/LiveFilter>. Figure 8.10 depicts an example of a RESTful Web service that allows querying the MIDEF ontology to get the number of design patterns instances presented in the MIDEF ontology. Particularly, 8.10-a and 8.10-b display the service results before and after running the GATE Java application, respectively. As displayed in Figure 8.10-b an additional design pattern is added in the MIDEF ontology. Consequently, the MIDEF ontology is successfully populated with "LiveFilter" design pattern instance.

- **IDEPAR and ICGDEP systems:** The MIDEF ontology is used in conjunction with the IDEPAR system and ICGDEP system described in Chapter 6 and Chapter 7, respectively. The IDEPAR system enables the design patterns represented in the MIDEF ontology to be automatically selected and retrieved. Whereas, the ICGDEP system allows the use of the MIDEF ontology for the generation of adaptive UIs using design patterns extracted from the developed ontology.

8.2. Evaluation of the MIDEP Ontology

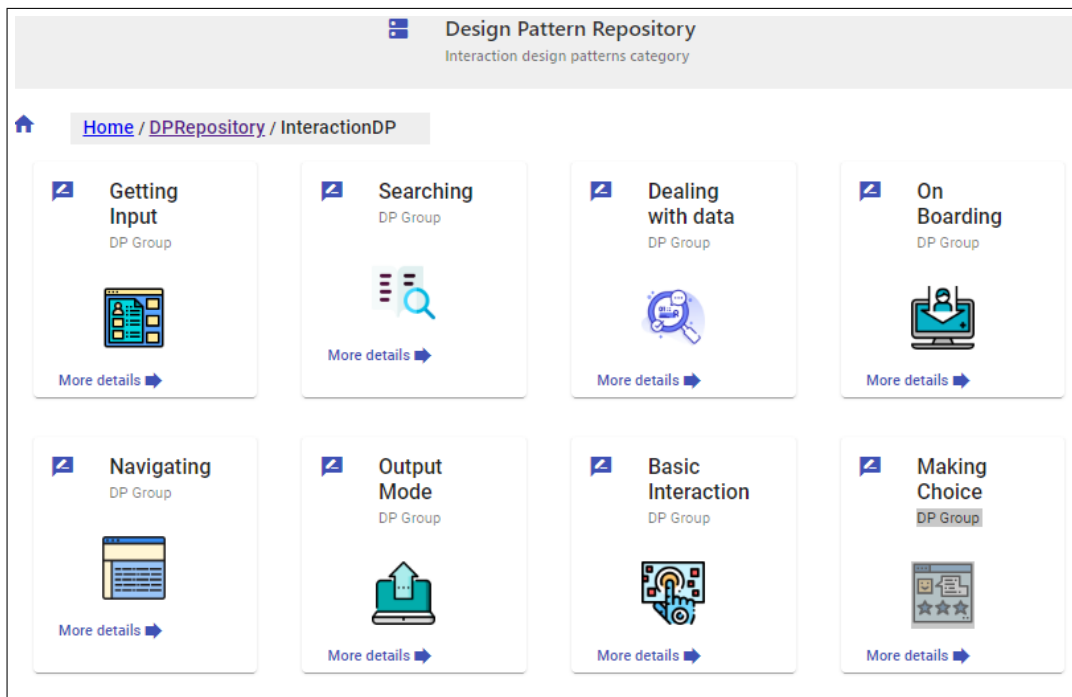


Figure 8.8: Interaction design pattern category

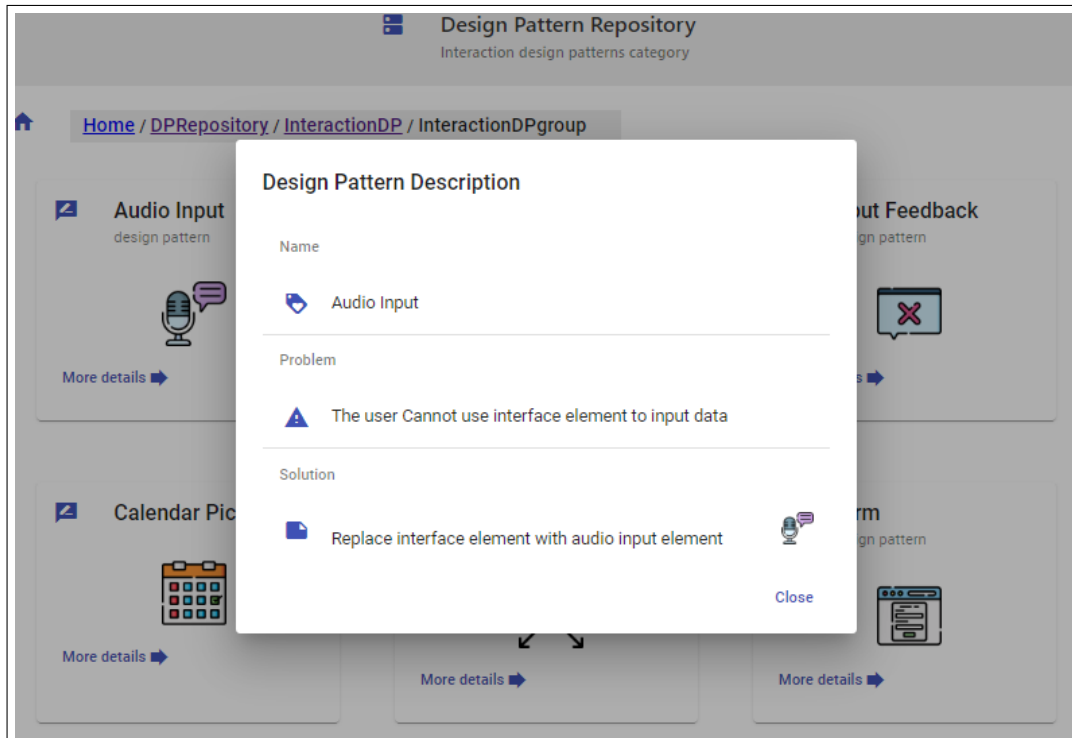


Figure 8.9: Example of design pattern description

The present evaluation approach shows how the MIDEP ontology can concretely be used within a variety of applications and systems.

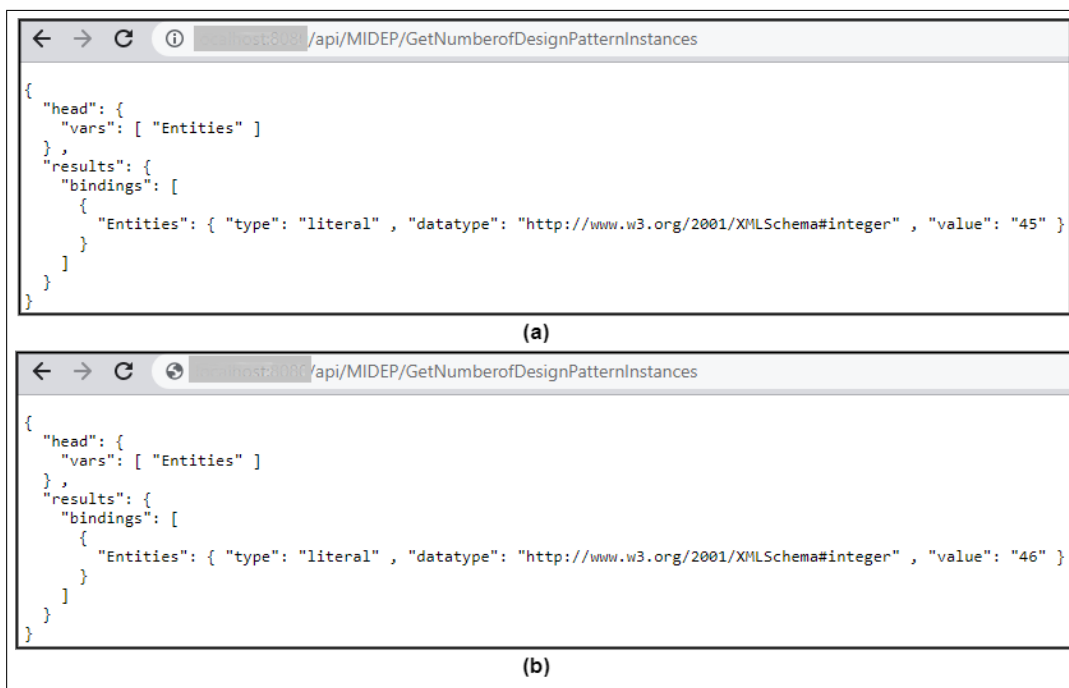


Figure 8.10: RESTful Web service result: (a) before ontology population, (b) after ontology population

8.2.5 Discussion

A three-fold approach to evaluate the MIDEP ontology is provided in the present section. First, the ontology is evaluated against the competency questions defined in the ORSD. Then, the OOPS! tool is applied to detect pitfalls and to ensure the correctness of the developed ontology considering different evaluation criteria. Finally, an application-based evaluation approach is provided to verify the use of the developed ontology in concrete applications.

The conducted evaluation revealed that the developed MIDEP ontology is: (i) correct since it meets the completeness, conciseness, and consistency standards, and (ii) effective since it can be used concretely in a variety of applications and systems.

8.3 Evaluation of the IDEPAR System

We have conducted two different experimental studies in order to achieve a comprehensive evaluation of the proposed IDEPAR system regarding various relevant dimensions. To that purpose, we performed two evaluation studies in different settings. The first study is used to evaluate the system's performance against an expert-based gold standard. The second one is a user-centric evaluation study that aims to assess the user experience with the proposed system, and the relevance of the recommended HCI design patterns. The main purpose of the two experimental studies, considered in the present section, is to test the performance of the proposed recommender system and to figure out the impact of the recommendations on participants' acceptance intentions regarding the IDEPAR system. In this regard, the following hypothesis are formulated:

- H1: The proposed IDEPAR system is efficient and performs well.
- H2: Participants' perceived experience of the IDEPAR system is positive.
- H3: The recommended HCI design patterns are relevant and match well with the given design problem.

8.3.1 Study 1: Expert-based gold standard evaluation

In order to verify the hypothesis H1, presented in Section 8.3, we created in this first study an expert gold standard allowing an expert to select and identify HCI design patterns that match with the given design problems. The objective of this study is to compare the gold standard expert recommendations with those retrieved by the IDEPAR system.

8.3.1.1 Study design

We invited an expert in the HCI domain and asked him to build a sample HCI design pattern on some subset of specific problem scenarios. To set up the present experiment and create the gold standard that we can compare our recommender system against the gold standard, we provided the expert with the following data:

- Design problem scenarios: we created a corpus of design problems composed of a set of design problem scenarios. The scenarios represent a description of user interface issues, user profile, etc. This corpus was given to the expert, invited in this gold based evaluation study, to choose the appropriate design pattern for each design problem scenario.
- HCI design patterns: for each design problem scenario, the expert is asked to select the relevant HCI design problem. To this purpose, we provided the expert a collection of HCI design patterns presented in Chapter 5. This collection includes HCI design patterns with their descriptions, including the name, the problem, the solution, the design pattern category and group. The design patterns presented in the collection are those included in the semantic knowledge base of the IDEPAR system. The HCI design patterns selected by the expert were included in the gold standard. Hence, by applying the proposed IDEPAR system with the same problem scenarios it is possible to compare the recommendations gained by an expert against those obtained automatically by the IDEPAR system.

8.3.1.2 Evaluation metrics

In this experiment, we adopt three performance measures, including precision, recall, and F-Measure [Powers, 2020]. When comparing the results of the IDEPAR system to the expert gold standard, a design pattern occurrence can be either a True Positive (TP), a False Positive (FP), a True Negative (TN), or a False Negative (FN). In the sense of this experiment, the set of TP includes all pattern occurrences recommended by the proposed system and presented in the gold standard. The set of FP contains all pattern occurrences recommended by the IDEPAR system but not presented in the gold standard. The set of TN contains all pattern occurrences not recommended by the proposed system and not presented in the gold standard. The set of FN contains all pattern occurrences not

recommended by the proposed system but presented in the gold standard. On that base, precision is defined as how large is the fraction of TP in the recommended design patterns and is computed using Equation (2). Then, recall is defined as how large is the fraction of TP in all design patterns and is given by Equation (3). Finally, the F-Measure is defined as the harmonic mean between precision and recall [Damljanovic et al., 2012] and is computed using Equation (4).

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (3)$$

$$\text{F-Measure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

8.3.1.3 Performance results

The present evaluation study includes scenarios that focus on particular design problems, which address problems related to the UI design and interaction issues. Eight samples of these design problem scenarios were given to the expert and were considered for illustrating the performance of the IDEPAR system. Information regarding the given design problem scenarios are shown in Table 8.2. First, Table 8.2 presents the type of each design problem that could be atomic or composite: atomic problems refer to the smallest sub-design problems, while composite design problems refer to problems that can be decomposed into simpler ones. Second, Table 8.2 illustrates the category of each design problem that could be run-time or design-time: (i) Design-time category supports design problems that are faced by designer or developer during application coding. More specifically, the design-time category includes design problems that represent application requirements. (i i) Run-time category supports common design problems faced by end users when using the final application. In particular, the run-time category includes design problems that represent end user requirements.

According to the samples of design problem scenarios, described in Table 8.2, the expert selects the suitable design patterns from the repository of HCI design patterns. The gold standard provided by the expert is presented in Table 8.3, which illustrates the list of HCI design patterns for each design problem scenario.

We evaluated the performance of the IDEPAR system by using the performance measures. In particular, we calculated measures of precision, recall, and F-measure. Table 8.4, for each of the design problem scenarios, reports the number of HCI design patterns provided by the expert and presented as Gold Standard (GS), the number of HCI design patterns recommended by the IDEPAR system, the number of TP, the number of FP, the number of TN, and the number of FN. The last three columns illustrate, respectively, the precision, recall, and F-measure for the IDEPAR's recommendations as compared to the expert gold standard design patterns list. The obtained results, outlined in Table 8.4, show that the IDEPAR system achieves good precision, recall and F-measure scores. In particular, the precision and recall results ranged from 0.5 to 1, whereas the F-measure results ranged from 0.56 to 1. When comparing the IDEPAR recommendations with experts' recommendations as gold standard, a precision of 72%, recall of 86.37%, and F-measure of 75.37% were observed. Thus, recommendations provided by the IDEPAR system are deemed both accurate and precise.

Table 8.2: Overview of the design problem scenarios

Scenario	Type	Category	Description
Problem1	Composite	Run-time	The current user is color-blind and he is interested in searching and finding his points of interest.
Problem2	Atomic	Run-time	A user with low vision disability is presented to interact with the current application.
Problem3	Atomic	Design-time	The application displays interfaces with low-contrast.
Problem4	Atomic	Run-time	The current user has hearing disability and the application provides an interface that uses audio modality.
Problem5	Composite	Design-time	The designer of the interface desires users with low vision disability to be able to navigate on the map and to locate their specified points of interests on the map.
Problem6	Composite	Run-time	The user cannot perceive colors, The user needs to find the location of a point of interest
Problem7	Composite	Run-time	The user has cognitive disability, The user needs to schedule an event
Problem8	Composite	Run-time	The user has a low vision disability; The user needs to do online shopping.

Table 8.3: List of HCI design patterns selected by expert

Scenario	Selected Design Patterns
Problem1	ZoomIn, ZoomOut, MapNavigator, DarkBackground
Problem2	FontSizeLarge, ZoomIn, AudioInput
Problem3	LightBackground , DarkFontColor, WhiteFontColor, Dark-Background
Problem4	AudioOutputVolumeIncreasing
Problem5	ZoomIn, Zoom Out, Map Navigator, Font Size Large, Audio Input
Problem6	ZoomIn, ZoomOut, MapNavigator, DarkBackground
Problem7	Calendar Picker, GuidedTour, Gallery, ActionButton
Problem8	FontSizeLarge, ZoomIn, ShoppingCart, ProductComparison, Booking, ProductAdvisor

Table 8.4: Performance results

Scenario	GS	IDEPAR	TP	FP	TN	FN	Precision	Recall	F-measure
Problem1	4	5	4	1	40	0	0.8	1	0.88
Problem2	3	4	2	2	40	1	0.5	0.66	0.56
Problem3	4	2	2	0	41	2	1	0.5	0.66
Problem4	1	2	1	1	43	0	0.5	1	0.66
Problem5	5	6	5	0	39	0	1	1	1
Problem6	4	5	4	1	40	0	0.80	1	0.88
Problem7	4	7	3	3	37	1	0.50	0.75	0.60
Problem8	6	9	6	3	36	0	0.66	1	0.79

8.3.2 Study 2: User-centric evaluation

In order to test hypothesis H2 and H3, we propose in the second experiment a user-centric evaluation study. The objective of this study is to figure out the impact of the recommendations on the participants' acceptance intention towards the proposed IDEPAR system.

8.3.2.1 Study design

In this study, users from different sources with a minimum experience in the HCI domain were invited via mailing lists. Among the participants, 33% were male and 67% were female with the majority being aged between 25 and 40 years (75%). Regarding the participants' academic discipline, this study was conducted on researchers (58%), software developers (33%), and computer science students (9%). After having accepted the invitation, participants were informed about the tasks required for this evaluation study. At first, they were given a guide describing how to use the proposed IDEPAR system. Then, they were asked to access the application developed to test the recommender system.

8.3.2.2 Study protocol

To verify the previously mentioned hypotheses (H2, H3), participants were asked to carry out two main tasks. The first task was to fill the pre-study questionnaire, whereas the second one was focused on answering the post-test questionnaire. More specifically, the pre-study questionnaire was oriented towards gathering participants' information regarding their knowledge about recommender systems, as well as their level of expertise with HCI design patterns. Concerning the post-test questionnaire, it was mainly considered to evaluate the quality of user experience with the IDEPAR system and the relevance of the recommended design patterns. This questionnaire was prepared based on the ResQue framework, which is a well-known user-centric evaluation recommender system for assessing user's experience and their acceptance [Pu et al., 2011]. The ResQue framework presents a wide variety of question statements that are categorized into four layers, including perceived system quality, belief, attitude and behavioral intention. A brief description of each evaluation layer is provided below.

- Perceived system quality: this layer includes questions, which assess participants' perception of the objective characteristic related to the recommender system.

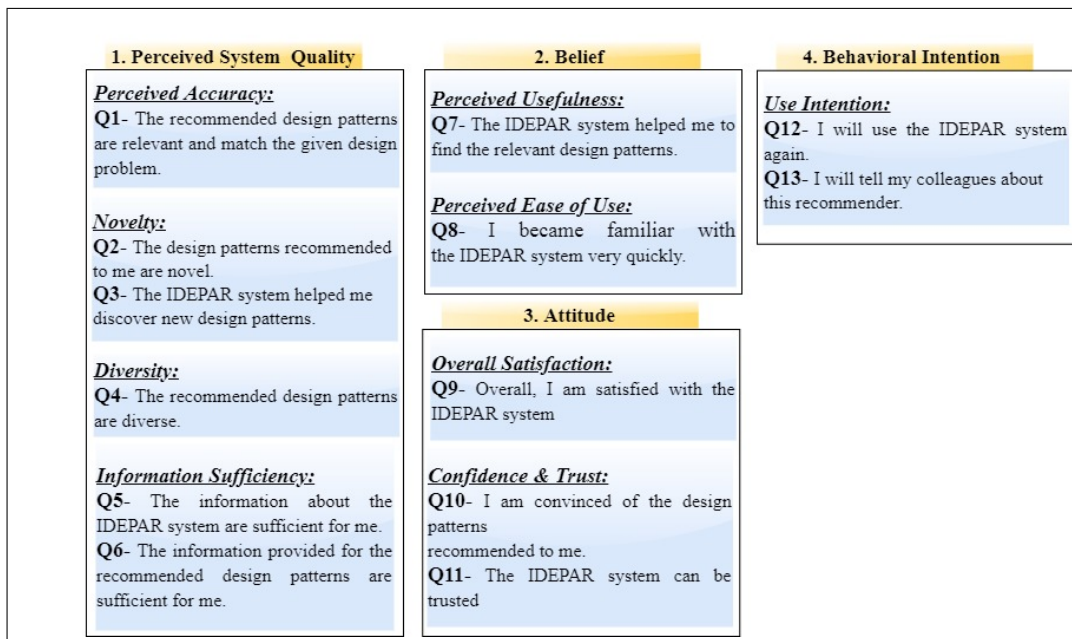


Figure 8.11: Representative questions from each ResQue layer

- Belief: this layer refers to questions that assess a higher level of participants' perception of the recommender system.
- Attitude: this layer includes questions, which assess participant's overall feeling regarding the recommender system.
- Behavioral intention: this layer refers to questions that assess the system's capability to engage participants to use it regularly.

Questions that belong to these layers mainly address participants' perceived experiences of the recommender system and accuracy of design patterns. Indeed, these questions answered the two hypotheses H2 and H3. In the post-test questionnaire, we consider 13 questions from the ResQue questionnaire and use the five point Likert scale (range 1 to 5) as the measurement scale in order to assess the degree of participant's answers, with 1 signifying "strongly disagree", and 5 signifying "strongly agree". The selected questions and their categories are presented in Figure 8.11. The full version of the post-test questionnaire is available in Appendix 9.5.

8.3.2.3 Statistical Analysis

In this evaluation study, we used IBM SPSS version 28.0 [SPSS, 2013] in order to perform the statistical analysis of data collected from the pre-study and post-test questionnaire. Descriptive analysis was substituted for the collected data. Particularly, measures of frequency (percent), central tendency (mean) and measures of dispersion (Standard Deviation (SD)) were considered. In addition, the reliability of the post-test questionnaire's layers were assessed using Cronbach's alpha [Anthony Jnr, 2021]. Finally, Pearson correlation was used to find the correlation between the experience level of participants and their answers. To test the correlation, a p-value of ≤ 0.05 was considered to be statistically significant.

Table 8.5: Pre-study questionnaire results

Questionnaire	Item	Percent
Knowledge of recommender system	Low	16%
	Medium	42%
	High	42%
Level of expertise with HCI design patterns	Novice	8%
	Intermediate	33%
	Advanced	59%

8.3.2.4 Pre-Study Questionnaire Results

A total of 12 participants were involved in the user-centric evaluation study, and answered the pre-study questionnaire. The responses to the demographic data of the pre-study questionnaire were as follows: 16% of participants were not familiar with recommender systems, whereas the remaining participants possessed medium (42%) or high (42%) knowledge about recommender systems. Although there were a minority of participants who had low knowledge about recommender systems (16%), the majority of participants had a medium or high level of knowledge (84%). Regarding the level of expertise with HCI design patterns, the majority of participants (more than 90%) had experience with HCI design patterns, wherein 8% were novice, 33% were intermediate, and 59% were advanced. Even though there were two groups: those who had no experience (8%) and those who had less than 2 years of experience (33%), experienced participants or those with more than five years of experience with HCI design patterns were the largest group (59%). Thus, participants with an advanced level of expertise with HCI design patterns outperformed those with less experience. Table 8.5 presents the descriptive statistics regarding the demographic data of the pre-study questionnaire.

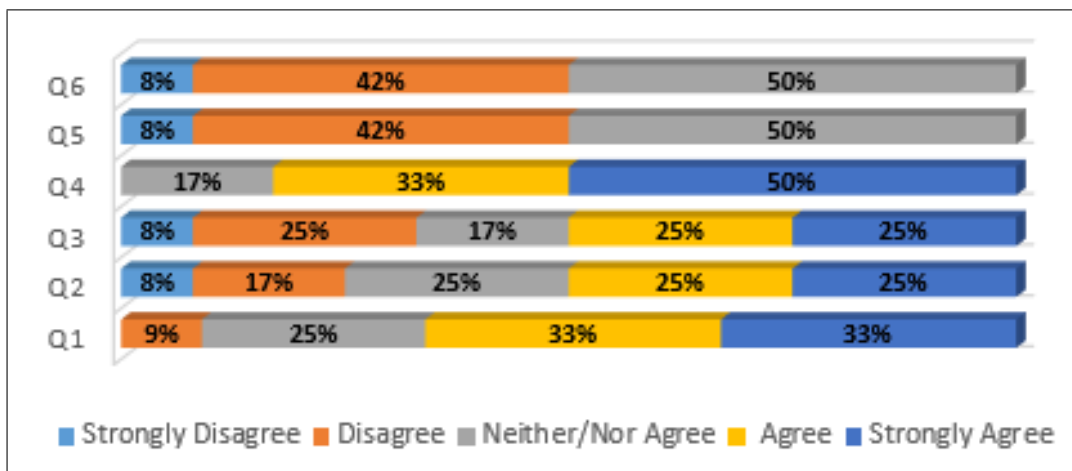
8.3.2.5 Post-Test Questionnaire Results

After filling the pre-test questionnaire, participants, who were involved in the user-centric evaluation study, answered the post-test questionnaire. Participant's answers were collected and analyzed. Table 8.6 illustrates descriptive statistics regarding the 13 questions of the post-test questionnaire. Along with the Cronbach alpha, other values were computed, including the mean, the SD, and the distribution of answers for each question item. Figures 8.12, 8.13, 8.14 and 8.15 display divergent stacked bars that show the distribution of participants' answers to the perceived system quality, belief, attitude, and behavioral intention layer, respectively.

According to the obtained results outlined in Table 8.6, we observed that the mean value for most of the questions was above the median, with SD values below 1. In particular, answers for the first question Q1, with a mean value of 3.91 (SD = 0.95), and the fourth question Q4, with a mean value of 4.33 (SD = 0.74), indicate that participants believed that the IDEPAR system recommended relevant and diverse HCI design patterns. For the second question Q2 and the third question Q3, roughly 50% of participants perceived the novelty of the recommended design patterns as shown in Figure 8.12. Meanwhile, answers of questions Q5 and Q6, amongst which are the information sufficiency questions, received the lowest scores comparing to other questions. Among all participants, 8% strongly disagreed and

Table 8.6: Post-test questionnaire results

Layer	Question	Mean	SD
Perceived System Quality	Q1	3.91	0.95
	Q2	3.41	1.25
	Q3	3.33	1.23
	Q4	4.33	0.74
	Q5	2.41	0.64
Belief	Q6	2.41	0.64
	Q7	3.91	0.95
	Q8	4.33	0.84
Attitude	Q9	3.41	1.25
	Q10	3.58	0.75
Behavioral Intention	Q11	3.83	0.68
	Q12	3.33	0.74
	Q13	3.16	0.68

**Figure 8.12:** Distribution of answers to post-test questionnaire: Perceived system quality layer

42% disagreed with questions Q6 and Q5. The mean value for these questions Q6 and Q5 was equal to 2.41 (SD = 0.64). These results revealed that half of participants were not well-satisfied with the sufficiency of the information about the recommender system (Q5) and the information provided for the recommended design patterns (Q6). Furthermore, Figure 8.13 illustrates that a half of participants were satisfied with the information sufficiency, in which 8% of participants did not agree, 42% disagreed, and 50% strongly agreed with question Q8. Additionally, the obtained mean value for the belief layer was high, being equal to 4.12. Participants' answers of the belief layer show that more than 90% of all participants agreed on question Q7 involving 50% strongly agreeing for question Q7, stating that "the IDEPAR system helped them to find the relevant design patterns". Regarding question Q8, participants' answers varied between 58% agree and 25% strongly agree. These results indicate that more than 80% of participants considered the proposed recommender system as useful, whereas a minority of them did not perceive the system as useful. For the attitude layer, answers of question Q9 show that participants' overall satisfaction was high with

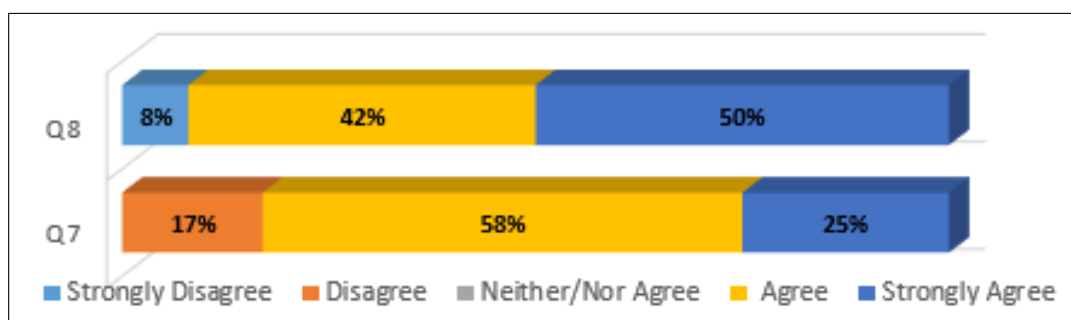


Figure 8.13: Distribution of answers to post-test questionnaire: Belief layer

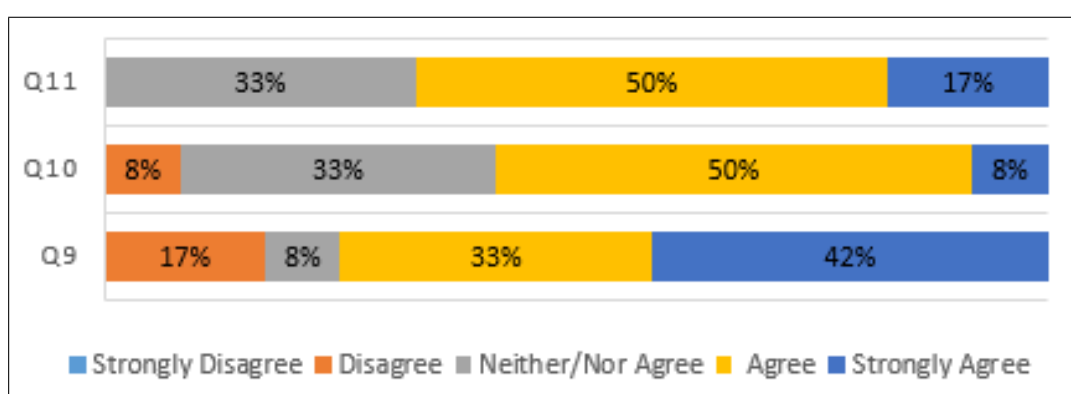


Figure 8.14: Distribution of answers to post-test questionnaire: Attitude layer

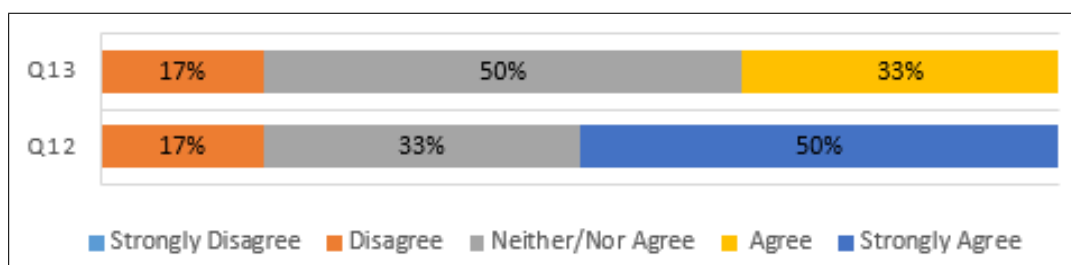


Figure 8.15: Distribution of answers to post-test questionnaire: Behavior layer

a mean value equal to 4 (SD = 1.08). As illustrated in Figure 8.14, most participants (60%) were satisfied with the recommender system. Moreover, the mean values for questions Q10 and Q11 were equal to 3.58 (SD = 0.75) and 3.83 (SD = 0.68), respectively. Finally, the mean value was equal to 3.24 (SD = 0.71) for the behavioral intention layer. This value reveals that participants found the proposed system moderately acceptable in terms of use intentions. Particularly, the mean values for Q12 and Q13 were equal to 3.33 (SD = 0.74) and 3.16 (SD = 0.68), respectively. As shown in Figure 8.15, participants' responses vary between 33% neither/nor agree and 17% disagree for question Q12, and 50% neither/nor agree and 17% disagree for question Q13. These results indicate that a minority of participants were satisfied with the use intention.

In order to verify whether the internal consistency test provided reliable results or not, the Cronbach's alpha criterion was considered. This criterion has to meet a minimum threshold of 0.7 [Nunnally, 1994]. As illustrated in Figure 8.16, the results of the measurements of Cronbach's alpha met the required minimum threshold for the three layers namely,

perceived system quality, attitude, and behavioral intention layers, except for the belief layer.

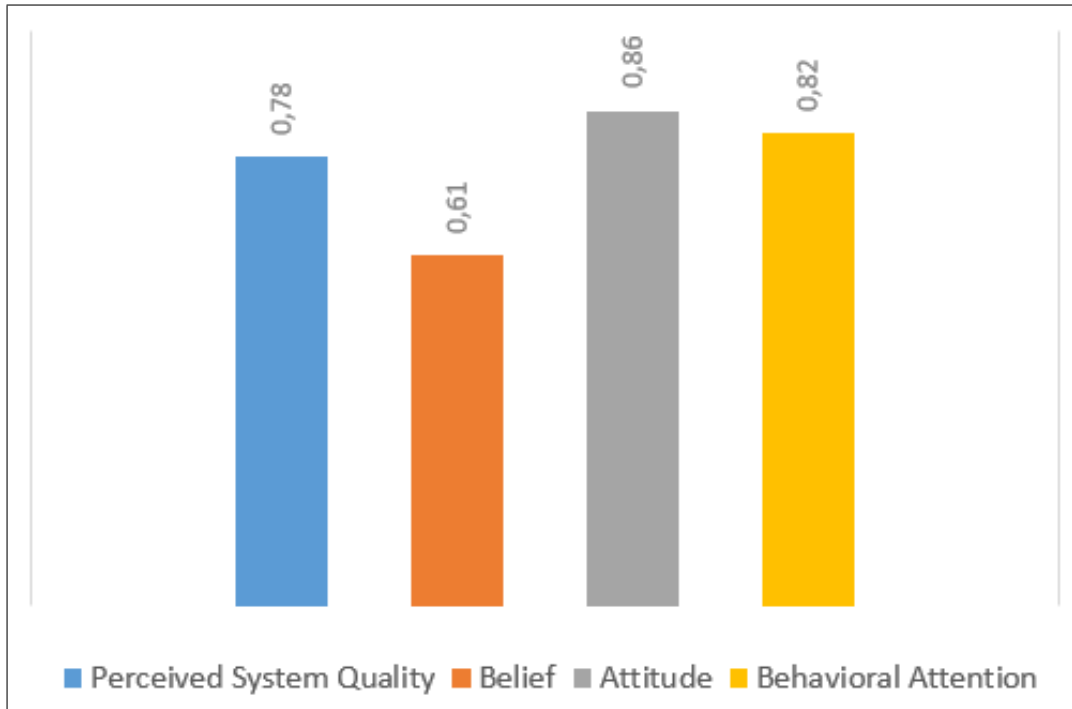


Figure 8.16: Results of the Cronbach's alpha coefficient

Besides, we investigated the correlation between the participants' expertise and answers using the Pearson's rank correlation coefficient. We relied on this coefficient since it provides values in the range from -1 to 1, which make it suitable for detecting negative correlations. Tables 8.7 and 8.8 present the correlations that were observed. In particular, Table 8.7 shows a significant correlation between participants' knowledge about recommender systems and the response of question Q7 ($r = 0.873$, $p < 0.001$) and question Q10 ($r = 0.711$, $p = 0.010$). Differently, Table 8.8 illustrates a significant correlation coefficient between participants' expertise with HCI design patterns and the answers of questions Q1 ($r = 0.080$, $p < 0.001$), Q7 ($r = 0.744$, $p = 0.005$), Q9 ($r = 0.598$, $p = 0.040$), and Q10 ($r = 0.595$, $p = 0.041$). The Pearson coefficient results show that participants, with good knowledge and experience with recommender systems and design patterns, found that the proposed IDEPAR system was helpful for retrieving relevant HCI design patterns ($p < 0.001$; $p = 0.005$) and confirmed that the recommended design patterns were convincing ($p = 0.005$; $p = 0.041$). Additionally, among experienced participants or those with more than five years of experience with HCI design patterns, the relevance of the recommended patterns ($p < 0.001$) and their satisfaction regarding the IDEPAR system ($p = 0.040$) was confirmed.

8.3.3 Discussion

In this section, we discuss the interpretation of the results obtained in the two presented experimental studies, including expert-based gold standard evaluation and user-centric evaluation.

Table 8.7: Correlations between participants' knowledge about recommender systems and answers of Q7 and Q10

	Q7	Q10
Pearson correlation	0.873 ¹	0.711 ¹
Sig. (1-tailed)	<0.001	0.010

^aCorrelation is significant at the 0.01 level (1-tailed).

Table 8.8: Correlations between participants' level of expertise with HCI design patterns and answers of Q1, Q7, Q9, and Q10

	Q1	Q7	Q9	Q10
Pearson correlation	0.880 ¹	0.744 ¹	0.598 ²	0.595 ²
Sig. (1-tailed)	<0.001	0.010		

^aCorrelation is significant at the 0.01 level (1-tailed).

^bCorrelation is significant at the 0.05 level (1-tailed).

Regarding the expert-based evaluation study, the performance results reveal that the proposed recommender system achieved high precision (76%), recall (83.20%), and F-measure (75.2%). Consequently, the IDEPAR system is deemed efficient and performs well. Concerning the user-centric evaluation study, the distribution of results for the pre-study questionnaire highlights the heterogeneous nature of participants who involve in the evaluation study and shows highly experienced participants. Then, answers of the post-test questionnaire show that the majority of participants (66%) confirmed that "the recommended design patterns are relevant and match with the given design problem". Furthermore, according to participants' responses to questions Q2 and Q3, 50% of participants agreed with the novelty of the proposed recommender system. Additionally, the overall mean of the belief layer was equal to 4.12 and exceeded the "Agree" value. Indeed, participants generally believed that the IDEPAR system helped them to find relevant HCI design patterns and perceived the ease of use of the provided system. For the attitude layer, the mean value was equal to 3.80 (SD = 0.83), which indicates the overall satisfaction of the participants and a high trust of the IDEPAR system. Regarding the behavioral intention layer, results show that 50% of participants strongly agreed that they would use the IDEPAR system again, and 30% of them agreed that they would recommend the system to their colleagues. Overall, we found that participants exhibited relatively low rates, especially for the information sufficiency and for the use intentions. These results may come from the difficulty of understanding the information provided by the recommender system. Thus, richer information concerning the recommended design patterns is needed. We consider this as a stimulus for the future enhancement of the IDEPAR system. Moreover, the reliability of the questions items was conducted with Cronbach's alpha. The obtained values for the perceived system quality, attitude, and behavioral intention layer exceeded the minimum threshold of 0.7, except for the belief layer. Consequently, the reliability was deemed good for the majority of layers and acceptable for only one layer.

In order to test the statistical significance of the correlation, the Pearson coefficient was applied for the target. In this sense, the obtained results reveal that knowledge of recommender systems and level of expertise with HCI design patterns appeared to be positively correlated with the answers of perceived usefulness, confidence, and trust. Further, the results of correlation analysis show that experience with HCI design patterns has a positive relationship with the perceived accuracy, as denoted with $p\text{-value} < 0.001$. Overall, the correlation analysis illustrates that several factors influence participant attitudes about perceived accuracy, perceived usefulness, satisfaction, confidence, and trust. These factors were mainly related to participants' knowledge about recommender systems and their level of experience with HCI design patterns.

To sum up, the research and development of the IDEPAR system, presented in Chapter 6, allow us to support the previously mentioned hypotheses, including H1, H2, and H3. More specifically, the findings of the present evaluation study reveal that (i) the IDEPAR system is efficient and performs well (H1), (ii) participants have a positive experience regarding the IDEPAR system's quality (H2), and finally (iii) participants' perceived accuracy of the recommended HCI design patterns is assessed positively (H3).

8.4 Evaluation of the ICGDEP System

The developed ICGDEPTool, presented in Chapter 7, was evaluated in terms of being effectively used by the developer, considering developers' productivity factor. This factor constitutes the main requirements for the ICGDEP system. The following hypothesis were put forward during this evaluation study:

- H4: The ICGDEP system automates the process of generating source code for UI views.
- H5: The ICGDEP system hastens the UI development process.

8.4.1 Experimental settings

An experiment was designed in which two software developers, having university degrees in Computer Science and experience in creating mobile applications using the Ionic framework, were invited to develop the UI views for the scenario example presented in Figure 8.17. The first developer (dev-1) was asked to implement the UIs without any tool support. Whereas, the second one (dev-2) was asked to build the required UIs using the ICGDEPTool. We conducted this evaluation study because we wanted to track developers' productivity factors. The influence on this factor was inspected by recording the UI development time, which is measured as the duration of time taken to develop the interfaces for the scenario depicted in Figure 8.17.

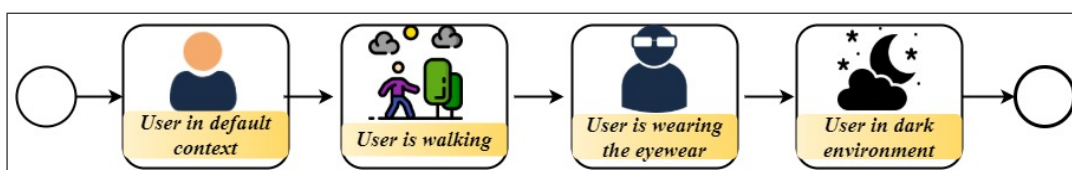


Figure 8.17: Scenario example

8.4.2 Results and discussion

The first developer (dev-1) implements the UIs manually from scratch, whereas the second one (dev-2) uses the ICGDEP tool to get the UI source code. Figure 8.18 shows screenshots of the UIs provided by dev-2. These interfaces are automatically generated by the ICGDEP tool. Thus, it is worthwhile to note the ability of the ICGDEP system to generate the source code of different UI views automatically.

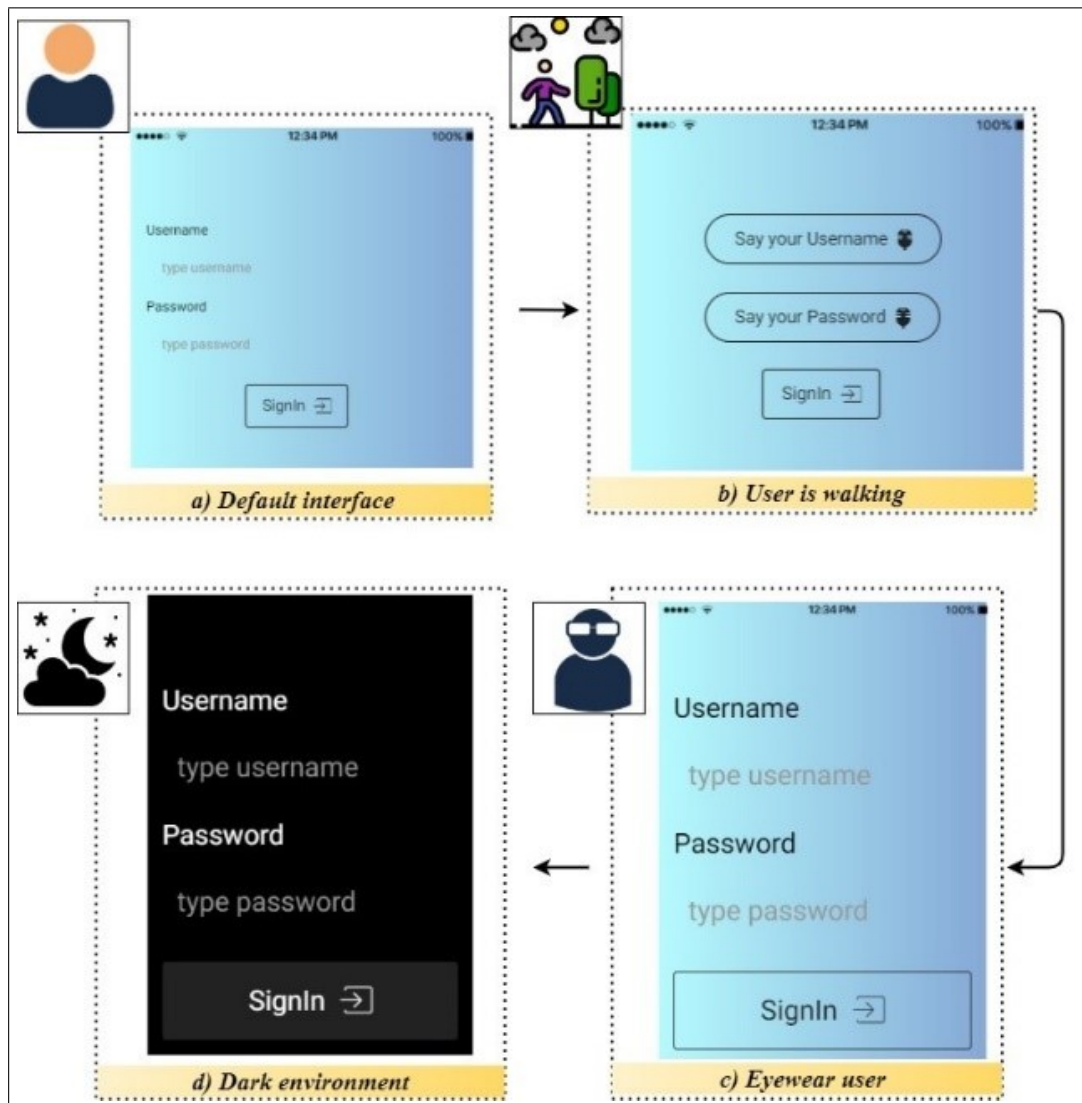


Figure 8.18: Screenshots of the generated user interfaces

The time spent by each developer to create the UIs were quite different among developers: the development time was 13 hours for "dev-1", whereas "dev-2", whose implementation method is based on the proposed tool, took only 2 hours. Hence, the obtained results show that the time spent is much longer in the case of "dev1", and is dramatically reduced for "dev-2". In general, these results indicate that the ICGDEP system helps to reduce the development time and has a quite good impact on increasing developer's productivity.

To sum up, the ICGDEP system we have introduced in Chapter 7 gives developers a

great deal of time and effort saving by providing mechanisms for the automatic generation of different UI views for Web and mobile applications. Consequently, the IDEPAR system allows us to support the previously mentioned hypotheses (H4 and H5). Particularly, the findings of the present evaluation study show that (i) the ICGDEP system automates the process of generating the source code for UI views (H4), and (ii) the ICGDEP system hastens the UI development process (H5).

8.5 Evaluation of the Global AUIDP Framework

The present study focuses on the evaluation of the global AUIDP framework introduced in Chapter 4 through a mobile application that was developed using the proposed framework. In particular, this study investigates the applicability of the AUIDP framework regarding two hypotheses with respect to the identified research questions in Chapter 1.

- H6: The AUIDP framework allows run-time UI adaptation by integrating the generated user interface source code in a running mobile or Web application.
- H7: The AUIDP framework generates usable interfaces that are accepted by end users.

This section can be broken down into the following: Subsection 8.5.1 describes the case studies with details of real world example scenarios that are based on the generated application. Next, Subsection 8.5.2 presents an evaluation study to prove the usability of the developed application. Final discussion regarding the previously described hypothesis (H6 and H7) is given in Subsection 8.5.3.

8.5.1 Case Studies

In this subsection, the global AUIDP framework is applied to different case studies in order to assess its capability to support the development of adaptive UIs automatically at run-time. In this sense, three application scenarios were selected to instantiate the AUIDP framework and to validate the hypothesis H6. These scenarios focused on the APA application that provides users various services according to their needs and abilities.

8.5.1.1 Case Study 1

As a first case study, we consider a user named Anna who has colorblindness. The present user is interested in visiting two points of interest (museum and hotel). After user registration and identification, information about the current user is sent to the AUIDP framework, which interacts with its systems, including the IDEPAR and the ICGDEP system. In this case study, the IDEPAR system automatically selects the following HCI design patterns that are relevant to the current situation: ZoomIn (DP1-1), ZoomOut (DP1-2), MapNavigator (DP1-3), DarkBackgroundColor (DP1-4), and WhiteFontColor (DP1-5). Using these design patterns recommended by the IDEPAR system, the ICGDEP system generates the UI source code for the current user. Accordingly, the global AUIDP framework automatically adapts the current UI for Anna. Figure 8.19 depicts the interface generated for this first case study.

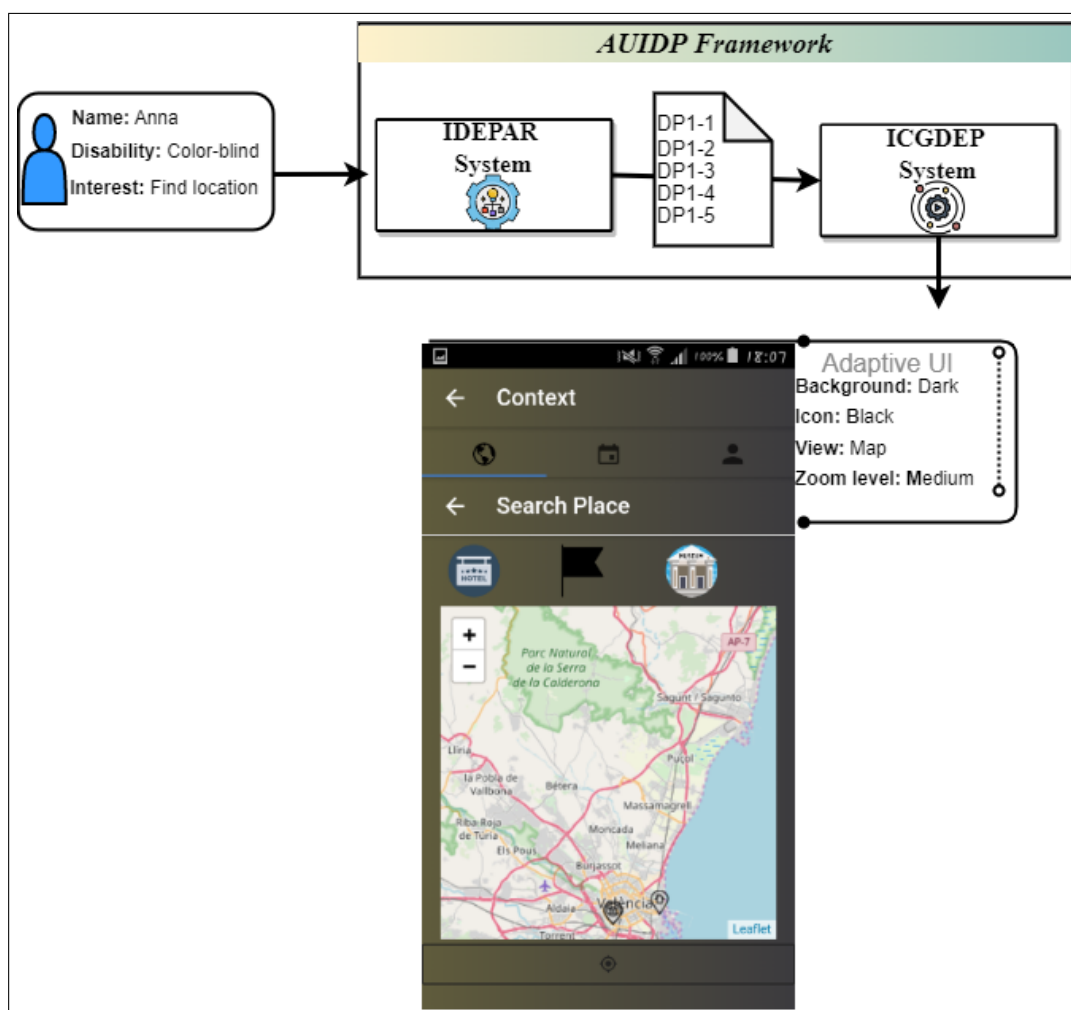


Figure 8.19: Case study 1

8.5.1.2 Case Study 2

In this second case study, Alex is a low vision user who is interested in finding a location of a point of interest (e.g., a hotel, a museum, or the nearest airport). Once connected, the AUIDP framework triggers the IDEPAR system. This latter recommends MapNavigator (DP2-1), ZoomIn (DP2-2), and FontSizeLarge (DP2-3) design patterns that are transferred to the ICGDEP system to create the final UI that includes interface elements appropriate to the current profile. Figure 8.20 shows the adaptive UI generated in this second case study.

8.5.1.3 Case Study 3

In this third case study, we consider a user named David who wants identify to the APA application while he is walking. As David is walking, the HCI design patterns that are recommended by the IDEPAR system, are AudioInput (DP3-1) and SignIn (DP3-2). Consequently, the UI source code created by the ICGDEP system correspond to a login interface with audio input modality. Figure 8.21 depicts the adaptive UIs generated by the global AUIDP framework for this third case study.

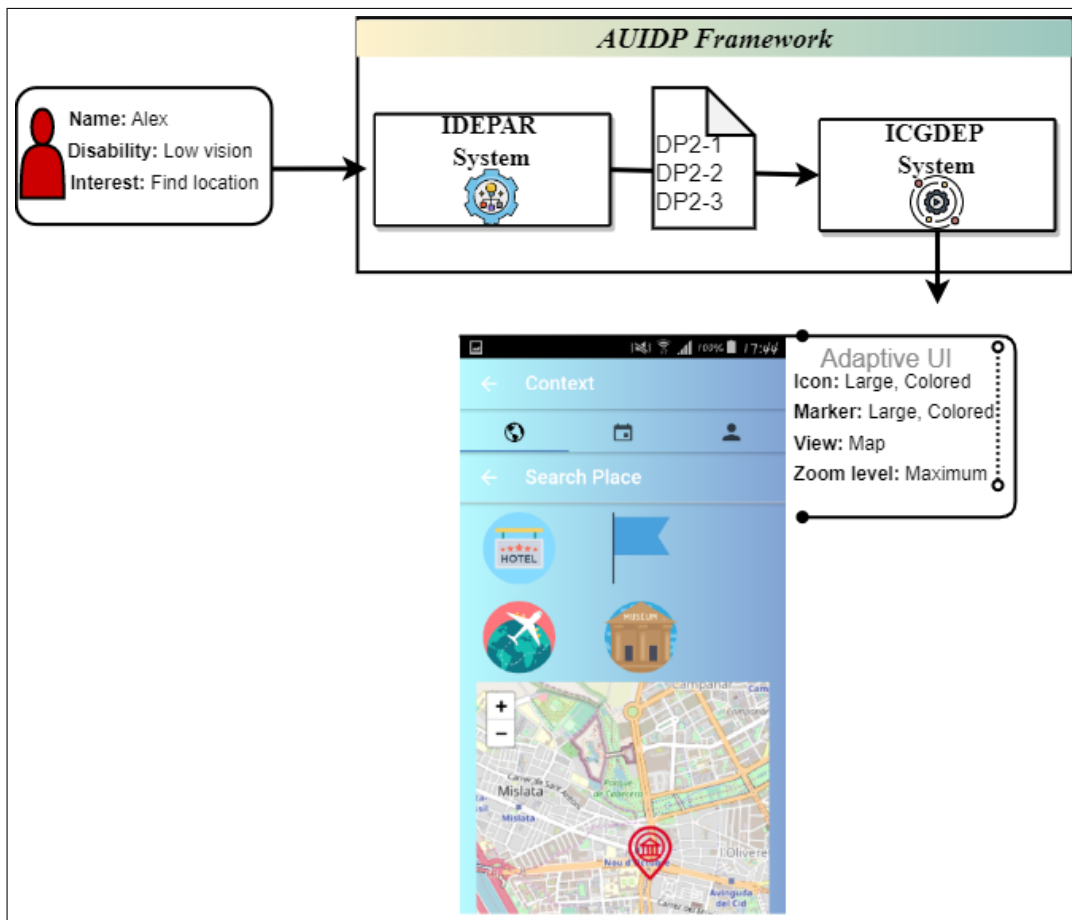


Figure 8.20: Case study 2

8.5.2 Usability study

The current evaluation employed a usability study of the APA application, previously presented in Subsection 8.5.1, to assess the usability of the interfaces that are designed and generated by the proposed AUIDP framework. The ultimate concern of the present study is to examine the following usability dimensions introduced by Nielsen [Nielsen, 1992]: (i) Learnability: refers to how well participants could work with the application. (ii) Satisfaction: refers to how pleasant participants' experience with the application was. (iii) Efficiency: refers to how long it took for a participant to complete a task.

To carry out these dimensions, the evaluation was done using two main instruments:

- First, the evaluation was conducted with an instrument of a usability questionnaire to elicit the satisfaction and learnability dimensions. The usability questionnaire was reproduced from the System Usability Scale (SUS) questionnaire [Brooke, 1996] and the Questionnaire for User Interaction Satisfaction (QUIS) [Norman et al., 1998]. The usability testing questionnaire is given in Appendix 9.5. The present questionnaire is divided into two main parts according to the satisfaction and learnability dimensions. It consists of 8 questions in which the satisfaction is defined by 5 items (Q'1,Q'2,Q'3,Q'4,Q'5) and the learnability is described through 3 items (Q'6,Q'7,Q'8). As outlined in Appendix 9.5, each question has to be answered using a scoring scale

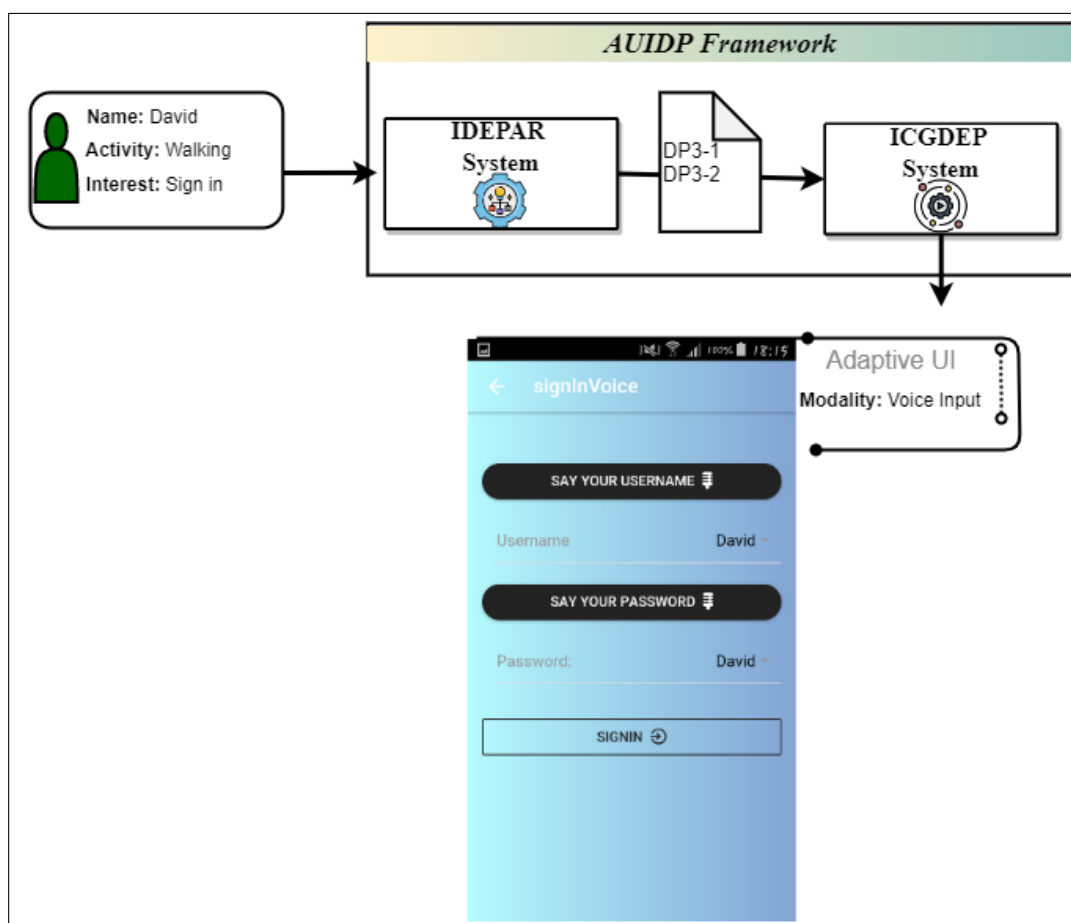


Figure 8.21: Case study 3

ranged from 0 to 4 (0 = To a very small extent, 1= To a small extent, 2 = Somewhat, 3 = To a large extent, 4 = To a very large extent). In this first instrument, descriptive analysis was substituted for the collected data. Specifically, measures of central tendency (mean) and measures of dispersion (SD) were considered. In addition, the reliability of the questionnaire's dimensions were assessed using Cronbach's alpha. In order to perform the analysis of the data gathered through the previously introduced questionnaire, the IBM SPSS version 28.0 [SPSS, 2013] software tool has been utilized.

- Second, we opted for measuring the Task Completion Time (TCT) to determine the efficiency dimension. The TCT was calculated by subtracting the end time from the start time of a task in two different interfaces that allow scheduling an event. The first interface was a traditional UI (Figure 8.22-a), whereas the second one was an adaptive UI generated by the proposed AUIDP framework (Figure 8.22-b). These interfaces propose two methods for scheduling an event that differ in their complexity. We were especially interested in whether the second interface, which is automatically created by the AUIDP framework, would be faster than the first interface, which is manually created from scratch.

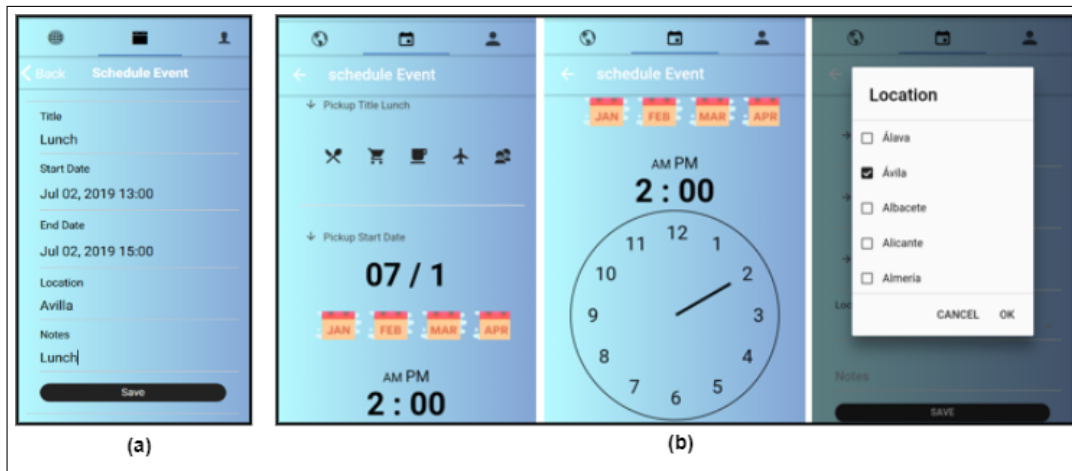


Figure 8.22: Interfaces for event scheduling: (a) Traditional interface, (b) Adaptive interface

8.5.2.1 Participants

This evaluation study involved the participation of eight postgraduate students of a master's degree program in Informatics Engineering from the Polytechnic University of Valencia (UPV) and the Higher Institute of Applied Science and Technology of Sousse (ISSATSO). The students that conducted this evaluation participated as end users. These students had similar background and different type of interests and disabilities. Among them, 75% were female and 25% were male with the majority being aged between 25 and 32 years (87.5%).

8.5.2.2 Results

The descriptive statistics regarding the eight questions of the usability questionnaire are given in Table 8.9. For the first question (Q'1), the mean value is 2.75 with an SD value equal to 0.82. These results indicate that user satisfaction for the applications in terms of easy to use is above the average. Moreover, the readability of texts (Q'2) reached the top score. In particular, the obtained mean value is equal to 4 with a null SD value which confirms that all participants agrees that texts on the interfaces are readable. Participants were also asked about the application color display (Q'3) and the size of the interface elements (Q'4). Answers for question Q'3, with a mean value of 2.62 (SD = 1.11), and question Q4, with a mean value of 2.87 (SD = 0.78), reveal that participants believed that the size of interface items is suitable and fit perfectly with their needs. The matching between the provided UIs with their needs (Q'5) received a medium score with a mean value of 2.37 (SD = 1.11). Questions with the highest scores were presented in the learnability dimension (Q'6–Q'8). These results confirmed that the application tasks could be performed in a straightforward manner, and the time to learn the application was estimated as short. In addition to mean and SD measures, alpha Cronbach's value (α) was calculated. As shown in Table 5, the coefficient Cronbach's alpha for satisfaction dimension was equal to 0.71. This value is considered acceptable, which means that participants are generally satisfied with the application. Concerning the second dimension, the alpha Cronbach's value for learnability is the highest, with a value of 0.91. Thus, meaning that the learnability was excellent for participants. Regarding the overall alpha Cronbach's value, the obtained result was equal to 0.87, which confirms the good internal consistency of the questionnaire.

Table 8.9: Statistical results of satisfaction and learnability dimensions

Usability Dimension	Questions	Mean	SD	Cronb. coeff
Satisfaction	Q'1	2.75	0.82	0.71
	Q'2	4	0	
	Q'3	2.62	1.11	
	Q'4	2.87	0.78	
	Q'5	2.37	1.11	
Learnability	Q'6	3.25	0.96	0.91
	Q'7	2.62	1.11	
	Q'8	3.25	1.08	

After filling the questionnaire, participants were invited to schedule events using traditional and adaptive interfaces shown in Figure 8.22. In this context, the average completion time across participants was computed. In the first case, the TCT average was about 102s, whereas in the second case the average time lowers to 45s. These results illustrate that the adaptive UI got shorter task scheduling completion time and better results for the average TCT. The obtained results reveal that participants completed their task quickly (50% of saved time) using the adaptive interface. Consequently, the adaptive interface, created by the AUIDP framework, reduces TCT and facilitates the way participants schedule an event.

8.5.3 Discussion

The case studies, introduced in the first evaluation study, demonstrates the capability of the proposed AUIDP framework to support the development of adaptive UIs. In particular, by using the AUIDP framework, adaptive UIs for the presented application scenarios are automatically generated at run-time. Moreover, based on the result interpretations of the usability study, we can conclude that the interfaces, created using the AUIDP framework, has a quite good usability in term of efficiency and learnability, and an acceptable usability in term of satisfaction. Consequently, the use of the AUIDP framework to design and generate adaptive UIs is effective.

To sum up, the proposed AUIDP framework was helpful for addressing hypothesis H6 and H7 as it: (i) allows run-time UI adaptation by integrating the generated UI source code in a running application (H6) and (i i) enables the automatic generation of usable UIs that are accepted by end users (H7).

8.6 Concluding Remarks

This chapter describes the evaluations conducted to validate the work presented in this thesis. Specifically, this work has been validated through different perspectives with respect to the confronted research questions introduced in Chapter 1. The main conclusions are described as follows:

- First, according to the conducted evaluation study described in Section 8.2, the MIDEP

ontology meets the completeness, conciseness, and consistency standards, and can be used concretely in a variety of applications and systems. Consequently, the developed ontology is correct and effective.

- Second, the findings of the evaluation study, presented in Section 8.3, reveal that the IDEPAR system performs well and is associated with a positive perceived experience. Furthermore, the IDEPAR system is able to recommend relevant HCI design patterns that solve the given design problems.
- Then, the evaluation, described in Section 8.4, shows that the ICGDEP system automates the UI source code generation. In turn, it hastens the UI development process.
- Finally, the findings of the evaluation study, described in Section 8.5, demonstrates the capability of the proposed AUIDP framework to support the development of adaptive UIs at run-time. In addition, according to the result interpretations of the usability study, in terms of efficiency, learnability, and satisfaction dimensions, the AUIDP framework is able to carry out usable interfaces.

Conclusions and Future Works

9.1 Introduction

The present thesis has introduced an approach that aims at facing the challenge of designing and generating UIs using HCI design patterns. As outlined in each part of this document, different and innovative contributions were produced from the present work. In addition, this thesis can be improved and extended in various interesting research directions.

This chapter introduces the conclusions of the work developed in the present thesis. In this concluding chapter, we first summarize the main contributions in Section 9.2. In Section 9.3, we then provide an overview of the publications that have emerged from this thesis. After that, Section 9.4 presents a future outlook on research directions and open issues that can extend the present research line. Finally, Section 9.5 concludes this chapter by summarizing the benefits and limitations of the present work.

9.2 Summary of Contributions

This thesis is part of a long-term research work to provide an approach for the design and generation of UIs for Web and mobile applications based on expert knowledge. The main contribution of this thesis is the AUIDP framework, which is proposed for designing and generating adaptive UIs by using design patterns. The present framework encompasses various contributions that are made towards achieving the research goals and answering the established research questions. In the following, a brief description of the contributions of this thesis is provided:

- **AUIDP framework:** A framework has been defined to support the design and generation of adaptive interfaces using expert knowledge. This framework introduces methods and tools from design-time to run-time for achieving automatic UI adaptation. One of the central features of the AUIDP framework is the ability to automatically develop Web and mobile applications using HCI design patterns that incarnate expert knowledge. In this sense, the proposed framework provides new capabilities for integrating the HCI design patterns for the development of adaptive

UIs. Consequently, the AUIDP framework brings a solution for representing knowledge related to HCI design patterns, for selecting appropriate HCI design patterns, and for generating adaptive UIs at run-time.

- **Method for design pattern specification:** A specification method has been defined to establish an ontology model that turns traditional text-based representation into formal HCI design patterns representation. The transition from informal to formal representation is achieved by applying the Neon methodology. The resulting MIDEP ontology is exploited at run-time to derive the HCI design patterns.
- **IDEPAR system:** A HCI design pattern recommender system has been defined in order to select the most relevant design patterns for a specific design problem. To process the recommendation requests received from developers and designers, a Web application has been developed that displays the list of HCI design patterns selected by the IDEPAR system to solve the given design problems.
- **ICGDEP system:** A generation system has been defined to enable the automatic generation of different UI views in a fully automatic fashion, starting with pattern instantiation, going through pattern integration, and ending with source code generation. In this context, a tool has been developed to provide UI views for Web and mobile applications that are generated by the ICGDEP system.

Regarding the validation of the above described contributions with respect to the confronted research questions, we have demonstrated the following facts as outlined in Chapter 8: First, the MIDEP ontology is correct and effective to represent knowledge related to HCI design patterns. Second, the IDEPAR system is efficient and performs well, as it promotes a positive user experience and it enables the recommendation of relevant HCI design patterns that solve the given design problem. Third, the ICGDEP system is feasible to automate the UI source code generation and to hasten the development process. Finally, the global AUIDP framework is able to carry out run-time UI adaptations and to generate usable interfaces that are accepted by end users.

9.3 Scientific Results

The research activity introduced in the present work has produced innovative and different contributions that have been published and discussed on different forums. The publications that emerged during the development of this PhD thesis allowed us to validate the contributions described in this work by the scientific community. The different publications are listed below. Under each publication, we describe the relevance of the forum where it was published.

- *International Journals Indexed in the JCR*
 - **Braham, A., Khemaja, M., Buendía, F., & Gargouri, F. (2021).** A Hybrid Recommender System for HCI Design Pattern Recommendations. *Applied Sciences*, 11(22), 10776
 - * According to the JCR, the *Applied Sciences* journal is in the second quartile (Q2) of the “Engineering, Multidisciplinary” category. (Impact Factor: 2,679)

-
- **Braham, A.**, Buendía, F., Khemaja, M., & Gargouri, F. (2021). User interface design patterns and ontology models for adaptive mobile applications. *Personal and Ubiquitous Computing*, 1-17.
 - * According to the JCR, the *Personal and Ubiquitous Computing (PUC)* journal is in the second quartile (Q2) of the "Computer Science, Information System" category. (Impact Factor: 3,453)
 - *International Conference and Workshop Papers*
 - **Braham, A.**, Khemaja, M., Buendía, F., Gargouri, F. (2022). Towards a Model-Driven Ontology-Based Architecture for Generating Adaptive User Interfaces. In: Novais, P., Carneiro, J., Chamoso, P. (eds) *Ambient Intelligence – Software and Applications – 12th International Symposium on Ambient Intelligence. ISAmI 2021. Lecture Notes in Networks and Systems*, vol 483. Springer, Cham. https://doi.org/10.1007/978-3-031-06894-2_13
 - * The 12th International Symposium on Ambient Intelligence (ISAmI 2021): The ISAmI conference has a crucial role within the Ambient Intelligence community
 - * The conference paper is published by Springer (LNCS).
 - **Braham, A.**, Khemaja, M., Buendía, F., & Gargouri, F. (2021, September). User Interface Adaptation through Ontology Models and Code Generation. In *Iberoamerican Workshop on Human-Computer Interaction* (pp. 225-236). Springer, Cham.
 - * The VII Iberoamerican Conference of Human-Computer Interaction (HCI 2021): The HCI conference is a forum for discussing and learning topics related to the Human Computer Interaction domain.
 - * The conference paper is published by Springer (CCIS).
 - **Braham, A.**, Khemaja, M., Buendía, F., & Gargouri, F. (2020, November). UI design pattern selection process for the development of adaptive apps. In *The Thirteenth International Conference on Advances in Computer-Human Interactions ACHI*, Valencia, Spain (pp. 21-27).
 - * The Thirteenth International Conference on Advances in Computer-Human Interactions (ACHI 2020): The ACHI conference includes fundamentals about interfaces and models, and covers new challenging research topics.
 - * The ACHI is a Core C conference according to the CORE conference ranking.
 - **Braham, A.**, Buendía, F., Khemaja, M., & Gargouri, F. (2019). Generation of adaptive mobile applications based on design patterns for user interfaces. In *Multidisciplinary Digital Publishing Institute Proceedings* (Vol. 31, No. 1, p. 19).
 - * The 13th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI 2019). UCAmI is considered as a reference event in the Ubiquitous Computing and Ambient Intelligence domain.

- * The conference paper is published by Proceedings, which is An Open Access Journal from MDPI.
- **Braham, A.**, Buendía, F., Khemaja, M.,& Gargouri, F. (2019). Towards designing and generating user interfaces by using expert knowledge. In the 10th International Symposium on Ambient Intelligence. (Accepted article)
 - * The 10th International Symposium on Ambient Intelligence (ISAmI 2019): The ISAmI conference has a crucial role within the Ambient Intelligence community.
 - * The conference paper is accepted and not published.

The contributions introduced in the present thesis are supported by these publications. In order to illustrate the relevance of the publications that emerged during the development of this PhD thesis, we provide in Table 9.1 the publications with the contributions achieved.

Table 9.1: List of the contributions and publications achieved

Contribution	Publication
Contribution #1: The global AUIDP framework	IsamI 2019, PUC Journal 2020
Contribution #2: A Method for design pattern specification	PUC Journal 2020, ACHI 2020
Contribution #3: The IDEPAR system	Applied Sciences journal 2021
Contribution #4: The ICGDEP system	UCAmI 2019, ISAmI 2021, HCI 2021

9.4 Future Works

The work presented in this dissertation is not a closed work and there are various interesting research directions that can be considered to improve the proposal. Thus, the research described in this thesis can be enhanced and extended in several interesting directions. In this section, the possible directions for follow-up research are the following:

- We consider that it is important to extend the proposed AUIDP framework to include new categories of design patterns. In this way, the present framework will be able to integrate new design solutions that will be used to design and generate UIs. Therefore, we need to extend the developed MIDEP ontology with existing design patterns that belong to new categories.
- We plan to extend the AUIDP framework to design and generate augmented reality UIs within immersive virtual reality. The main idea is to analyze perceptual and multimodal interactions of end users and accordingly provide the corresponding UIs. To achieve this, studying and detecting user behavioral design patterns could be convenient in order to allow users to naturally interact with the virtual and augmented environment.

9.5 Concluding Remarks

The work presented in this thesis has introduced an approach to face the challenge of designing and generating adaptive UIs using HCI design patterns. Specifically, the contributions described in this thesis have the following benefits:

- The use of expert knowledge is considered through the specification of HCI design patterns.
- The modeling effort achieved at design-time is not only useful for the specification of design patterns, but also yields a rich semantic base for the generation of UIs during run-time.
- Time to develop Web and mobile applications is reduced when the UI source code is automatically generated.
- The support of dynamic UI adaptations to meet users' needs without the necessity of developer or designer's interventions.

In this thesis, we limited the use of specific categories of HCI design patterns and the target UIs. This leads to the fact that augmented reality UIs are not supported. These limitations suggest future works to extend the proposal with more complex design patterns and to develop augmented reality UIs within immersive virtual reality.

Appendices

HCI Design Pattern Catalog

This appendix provides the descriptions of the HCI design pattern catalog. In the present thesis, we consider a set of 45 HCI design patterns that are formalized in the MIDE ontology and available in the knowledge base.

An excerpt of our catalog of HCI design patterns that belongs to "FontColor", "Background", "FontSize", "InputMode", "Zoom", "Navigating", "OutputMode", "MakingChoice", and "Shopping" group is given respectively in the following Tables (1-10). The present catalog contains three columns, including Design Pattern Name, Design Pattern Problem, and Design Pattern Solution.

Table 1: Description of HCI design patterns for "FontColor" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
BlackandWhiteColorDP	User has difficulties perceiving images and text.	Display black and white images, and black text.
LightColorDP	User has difficulties perceiving font color.	Set light font color
DarkColorDP	User has difficulties perceiving font color.	Set dark font color
ColoredFontDP	User has difficulties perceiving font color	Use colored font.

Table 2: Description of HCI design patterns for "Background" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
DarkBackgroundColorDP	User has difficulties perceiving background color.	Set dark background
LightBackgroundColorDP	User has difficulties perceiving background color.	Set Light background.
ColoredBackgroundDP	User has difficulties perceiving background color	Use colored background.

Table 3: Description of HCI design patterns for "FontSize" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
DarkBackgroundColorDP	User has difficulties perceiving background color.	Set dark background User has Color Blindness Disability
LightBackgroundColorDP	User has difficulties perceiving background color.	Set Light background.
ColoredBackgroundDP	User has difficulties perceiving background color	Use colored background.

Table 4: Description of HCI design patterns for "InputMode" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
AudioInputDP	User cannot use interface element to input data.	Replace interface element with audio input element.
TextInputDP	User needs to type a text.	Display a text input interface element.

Table 5: Description of HCI design patterns for "BasicInteraction" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
GalleryDP	User has difficulties understanding text.	Image such as icons or pictures can support better user understanding.
GuidedTourDP	User needs to learn about how they can do something.	Show users how to do something in several interactive steps
ActionButtonDP	User need to be aware of the important actions.	Use button with the action 'verb' as part of the label.
SlidesShowDP	User want to view a series of images.	Display images with animated transitions between images.

Table 6: Description of HCI design patterns for "Zoom" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
ZoomIn	Different users need different zoom level	Increase the zoom level.
ZoomOut	Different users need different zoom level	Decrease the zoom level.

Table 7: Description of HCI design patterns for "Navigating" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
MapNavigatorDP	The user needs to find a location of choice on a map	Display map navigator element
NavigationTabDP	Content needs to be separated into sections and accessed using a flat navigation structure that gives a clear indication of current location	Display a horizontal bar contains the different sections or categories
MenuDP	The user needs to access the main navigation	Repeat the main navigation on the bottom of the page
BreadcrumbsDP	The user needs to know his location in the Website's hierarchical structure	Display the labels of the sections and provide links to higher levels
NotificationDP	The user wants to be informed about important updates and messages	Notify users about relevant and timely events

Table 8: Description of HCI design patterns for "OutputMode" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
AudioOutputDP	The user cannot recognize interface element to read text on it	Replace interface element with audio output element
VolumeIncreasingDP	The user cannot hear application instructions or feedback	Repeat the audio and increase the volume.

Table 9: Description of HCI design patterns for "MakingChoice" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
RatingDP	The user needs to rate something	Display rating and the option to rate it
DateSelectorDP	The user needs to select a date	Display date selector
LanguageSelectorDP	The user needs to select their preferred language	Ask users the desired language

Table 10: Description of HCI design patterns for "Shopping" group

Design Pattern Name	Design Pattern Problem	Design Pattern Solution
ShoppingCardDP	The user wants to buy a product	Introduce a shopping cart where users can put their products in before they actually purchase them
BookingDP	The user needs to book something	Display search for objects and allow users make booking
StoreLocatorDP	The user wants to find a store close to a specific location	Allow users to search for a store and show the results on a map
VirtualProductDisplayDP	The user wants to try a product	Allow users to interact virtually with the product
ProductComparisonDP	The user needs to compare similar products	Show a matrix of products and features
VirtualProductAdvisorDP	Users want advice on selecting the best product for them among a set of products	Display products based on constraints, preferences and needs of users

Post-test Questionnaire

Table 11: Questions of the Post-test Questionnaire

Question	1 ¹	2 ²	3 ³	4 ⁴	5 ⁵
Perceived System Quality					
Q1- The recommended design patterns are relevant and match the given design problem.					
Q2- The design patterns recommended to me are novel.					
Q3- The IDEPAR system helped me discover new design patterns.					
Q4- The recommended design patterns are diverse.					
Q5- The information about the IDEPAR system are sufficient for me.					
Q6- The information provided for the recommended design patterns are sufficient for me.					
Belief					
Q7- The IDEPAR system helped me to find the relevant design patterns.					
Q8- I became familiar with the IDEPAR system very quickly.					
Attitude					
Q9- Overall, I am satisfied with the IDEPAR system.					
Q10- I am convinced of the design patterns recommended to me.					
Q11- The IDEPAR system can be trusted.					
Behavioral Attention					
Q12- I will use the IDEPAR system again.					
Q13- I will tell my colleagues about this recommender.					

^aStrongly Disagree

^bDisagree

^cNeither/ Nor Agree

^dAgree

^eStrongly Agree

Usability Testing Questionnaire

Table 12: Questions of the Usability Testing Questionnaire

Question	1 ¹	2 ²	3 ³	4 ⁴	5 ⁵
Satisfaction					
Q' 1- This application is easy to use.					
Q' 2- The texts characters on the interface are readable.					
Q' 3- The color used in the application help to a better content visualizationdisplay.					
Q' 4- The size of interface items can be considered rightsuitable					
Q' 5- The interface items fits with my current context or needs.					
Q' 6- The application is easy to learn.					
Learnability					
Q' 7- Application tasks can be achieved in a straightforward manner.					
Q' 8- The time required for learning the application can be considered short.					

^aTo a very small extent

^bTo a small extent

^cSomewhat

^dTo a large extent

^eTo a very large extent

Bibliography

- [Abdelhedi and Bouassidar, 2018] Abdelhedi, K. and Bouassidar, N. (2018). An soa design patterns recommendation system based on ontology. In *International Conference on Intelligent Systems Design and Applications*, pages 1020–1030. Springer. See pages 30, 32, and 33.
- [ABIRResearch, 2013] ABIRResearch (2013). Mobile application revenue generation. <https://www.abiresearch.com/press/tablets-will-generate-35-of-this-years-25-billion-/>. See page 1.
- [Ahmed and Ashraf, 2007] Ahmed, S. and Ashraf, G. (2007). Model-based user interface engineering with design patterns. *Journal of Systems and Software*, 80(8):1408–1422. See pages 28, 29.
- [Akiki et al., 2014] Akiki, P. A., Bandara, A. K., and Yu, Y. (2014). Adaptive model-driven user interface development systems. *ACM Computing Surveys (CSUR)*, 47(1):1–33. See page 2.
- [Alexander et al., 1979] Alexander, C. et al. (1979). *The timeless way of building*, volume 1. New york: Oxford university press. See pages 12, 13.
- [Alexander et al., 1977] Alexander, C., Ishikawa, S., and Silverstein, M. (1977). A pattern language: towns, buildings, construction: Oxford university press. *New York*. See pages 11, 12.
- [Andemeskel and Semere, 2018] Andemeskel, F. and Semere, D. T. (2018). Model-based collaborative development of manufacturing and control systems. *International Journal of Industrial and Systems Engineering*, 28(4):433–450. See page 26.
- [Anthony Jnr, 2021] Anthony Jnr, B. (2021). A case-based reasoning recommender system for sustainable smart city development. *AI & SOCIETY*, 36(1):159–183. See page 119.
- [Bechhofer et al., 2001] Bechhofer, S., Horrocks, I., Goble, C., and Stevens, R. (2001). Oiled: a reason-able ontology editor for the semantic web. In *Annual Conference on Artificial Intelligence*, pages 396–408. Springer. See page 19.
- [Beck, 1987] Beck, K. (1987). Using pattern languages for object-oriented programs. <http://c2.com/doc/oopsla87.html>. See page 12.
- [Beckett and McBride, 2004] Beckett, D. and McBride, B. (2004). Rdf/xml syntax specification (revised). *W3C recommendation*, 10(2.3). See page 20.
- [Berti et al., 2004a] Berti, S., Correani, F., Mori, G., Paterno, F., and Santoro, C. (2004a). Teresa: a transformation-based environment for designing and developing multi-device interfaces. In *CHI'04 extended abstracts on Human factors in computing systems*, pages 793–794. See pages 26, 29.
- [Berti et al., 2004b] Berti, S., Correani, F., Paterno, F., and Santoro, C. (2004b). The teresa xml language for the description of interactive systems at multiple abstraction levels. In *Proceedings workshop on developing user interfaces with XML: advances on user interface description languages*, pages 103–110. See page 26.
- [Bishop, 2007] Bishop, J. (2007). *C# 3.0 Design Patterns: Use the Power of C# 3.0 to Solve Real-World Problems*. " O'Reilly Media, Inc.". See page 13.

- [Blomqvist and Sandkuhl, 2005] Blomqvist, E. and Sandkuhl, K. (2005). Patterns in ontology engineering: Classification of ontology patterns. In *ICEIS (3)*, pages 413–416. See page 15.
- [Borchers, 2008] Borchers, J. O. (2008). A pattern approach to interaction design. In *Cognition, Communication and Interaction*, pages 114–131. Springer. See pages 12, 34.
- [Bouraoui and Gharbi, 2019] Bouraoui, A. and Gharbi, I. (2019). Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations. *Science of computer programming*, 172:63–101. See pages 35, 37, and 38.
- [Braham et al., 2021a] Braham, A., Buendía, F., Khemaja, M., and Gargouri, F. (2021a). User interface design patterns and ontology models for adaptive mobile applications. *Personal and Ubiquitous Computing*, pages 1–17. See pages 41, 51.
- [Braham et al., 2020] Braham, A., Khemaja, M., Buendía, F., and Gargouri, F. (2020). Ui design pattern selection process for the development of adaptive apps. See pages 41, 51.
- [Braham et al., 2021b] Braham, A., Khemaja, M., Buendía, F., and Gargouri, F. (2021b). A hybrid recommender system for hci design pattern recommendations. *Applied Sciences*, 11(22):10776. See page 42.
- [Braham et al., 2021c] Braham, A., Khemaja, M., Buendía, F., and Gargouri, F. (2021c). User interface adaptation through ontology models and code generation. In *Iberoamerican Workshop on Human-Computer Interaction*, pages 225–236. Springer. See page 42.
- [Brambilla and Fraternali, 2014] Brambilla, M. and Fraternali, P. (2014). *Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML*. Morgan Kaufmann. See pages 3, 27, and 29.
- [Brooke, 1996] Brooke, J. (1996). Sus: a “quick and dirty” usability. *Usability evaluation in industry*, 189(3). See page 129.
- [Buschmann et al., 2007] Buschmann, F., Henney, K., and Schmidt, D. C. (2007). *Pattern-oriented software architecture, on patterns and pattern languages*. John wiley & sons. See page 13.
- [Calvary et al., 2003] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonck, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with computers*, 15(3):289–308. See pages 2, 26.
- [Celikkan and Bozoklar, 2019] Celikkan, U. and Bozoklar, D. (2019). A consolidated approach for design pattern recommendation. In *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pages 1–6. IEEE. See pages 31, 33.
- [Chen et al., 2005] Chen, H., Finin, T., and Joshi, A. (2005). The soupa ontology for pervasive computing. In *Ontologies for agents: Theory and experiences*, pages 233–258. Springer. See page 60.
- [Coplien,] Coplien, J. O. Software design patterns: common questions and answers. See page 13.
- [Coram and Lee, 1996] Coram, T. and Lee, J. (1996). Experiences: A pattern language for user interface design. In *Proc. of Joint Pattern Languages of Programs Conferences PLOP*, volume 96, pages 1–16. See page 13.
- [Damljanovic et al., 2012] Damljanovic, D., Stankovic, M., and Laublet, P. (2012). Linked data-based concept recommendation: Comparison of different methods in open innovation scenario. In *Extended Semantic Web Conference*, pages 24–38. Springer. See page 116.
- [Dandan et al., 2018] Dandan, R., Desprès, S., and Nobécourt, J. (2018). Oafe: An ontology for the description of elderly activities. In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 396–403. IEEE. See page 60.
- [de Bruijn et al., 2004] de Bruijn, J., Feier, C., Keller, U., Lara, R., Polleres, A., Predoiu, L., and Lausen, H. (2004). Wsmml reasoning implementation. See page 15.

- [De Oliveira et al., 2013] De Oliveira, K. M., Bacha, F., Mnasser, H., and Abed, M. (2013). Transportation ontology definition and application for the content personalization of user interfaces. *Expert Systems with Applications*, 40(8):3145–3159. See page 26.
- [Di Martino and Esposito, 2016] Di Martino, B. and Esposito, A. (2016). A rule-based procedure for automatic recognition of design patterns in uml diagrams. *Software: Practice and Experience*, 46(7):983–1007. See page 14.
- [Efftinge and Völter, 2006] Efftinge, S. and Völter, M. (2006). oaw xtext: A framework for textual dsls. In *Workshop on Modeling Symposium at Eclipse Summit*, volume 32. See page 96.
- [Efftinge and Zarnekow, 2011] Efftinge, S. and Zarnekow, S. (2011). Extending java-xtend: a new language for java developers. *PragPub, The Pragmatic Bookshelf*, 1(30):5–11. See page 98.
- [Engel and Martin, 2009] Engel, J. and Martin, C. (2009). Pamgis: a framework for pattern-based modeling and generation of interactive systems. In *International Conference on Human-Computer Interaction*, pages 826–835. Springer. See pages 28, 29.
- [Erl, 2007] Erl, T. (2007). *SOA principles of service design (the Prentice Hall service-oriented computing series from Thomas Erl)*. Prentice Hall PTR. See page 13.
- [Farquhar et al., 1997] Farquhar, A., Fikes, R., and Rice, J. (1997). The ontolingua server: A tool for collaborative ontology construction. *International journal of human-computer studies*, 46(6):707–727. See page 19.
- [Fensel et al., 2001] Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D. L., and Patel-Schneider, P. F. (2001). Oil: An ontology infrastructure for the semantic web. *IEEE intelligent systems*, 16(2):38–45. See page 15.
- [Fernández-López and Gómez-Pérez, 2002] Fernández-López, M. and Gómez-Pérez, A. (2002). The integration of ontoclean in webode. *CEUR Workshop Proceedings*. See page 21.
- [Fincher et al., 2003] Fincher, S., Finlay, J., Greene, S., Jones, L., Matchen, P., Thomas, J., and Molina, P. J. (2003). Perspectives on hci patterns: concepts and tools. In *CHI’03 extended abstracts on Human factors in computing systems*, pages 1044–1045. See page 14.
- [France et al., 2004] France, R. B., Kim, D.-K., Ghosh, S., and Song, E. (2004). A uml-based pattern specification technique. *IEEE transactions on Software Engineering*, 30(3):193–206. See page 14.
- [Gabriel, 1996] Gabriel, R. P. (1996). *Patterns of software*. Citeseer. See page 12.
- [Gadasin et al., 2020] Gadasin, D., Shvedov, A., and Koltsova, A. (2020). Cluster model for edge computing. In *2020 International Conference on Engineering Management of Communication and Technology (EMCTECH)*, pages 1–4. IEEE. See page 1.
- [Gajos et al., 2010] Gajos, K. Z., Weld, D. S., and Wobbrock, J. O. (2010). Automatically generating personalized user interfaces with supple. *Artificial Intelligence*, 174(12-13):910–950. See page 3.
- [Gamecho et al., 2015] Gamecho, B., Minón, R., Aizpurua, A., Cearreta, I., Arrue, M., Garay-Vitoria, N., and Abascal, J. (2015). Automatic generation of tailored accessible user interfaces for ubiquitous services. *IEEE Transactions on Human-Machine Systems*, 45(5):612–623. See pages 34, 37, and 38.
- [Gamma et al., 1995] Gamma, E., Helm, R., Johnson, R., Johnson, R. E., Vlissides, J., et al. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH. See pages 12, 13.
- [Gamma et al., 1993] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1993). Design patterns: Abstraction and reuse of object-oriented design. In *European Conference on Object-Oriented Programming*, pages 406–431. Springer. See page 3.
- [Gangemi et al., 2005] Gangemi, A., Catenacci, C., Ciaramita, M., and Lehmann, J. (2005). Ontology evaluation and validation: an integrated formal model for the quality diagnostic task. *On-line: http://www.loa-cnr.it/Files/OntoEval4OntoDev_Final.pdf*. See page 22.

BIBLIOGRAPHY

- [Gennari et al., 2003] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., and Tu, S. W. (2003). The evolution of protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies*, 58(1):89–123. See page 19.
- [Gomaa et al., 2005] Gomaa, M., Salah, A., and Rahman, S. (2005). Towards a better model based user interface development environment: A comprehensive survey. *Proceedings of MICS*, 5. See page 2.
- [Gómez-Pérez, 2004] Gómez-Pérez, A. (2004). Ontology evaluation. In *Handbook on ontologies*, pages 251–273. Springer. See pages 19, 22, and 52.
- [Gregor and Hevner, 2013] Gregor, S. and Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS quarterly*, pages 337–355. See page 6.
- [Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220. See page 15.
- [Gruber, 1995] Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928. See page 15.
- [Grüninger and Fox, 1995] Grüninger, M. and Fox, M. S. (1995). Methodology for the design and evaluation of ontologies. See pages 16, 21, and 106.
- [Guarino and Welty, 2002] Guarino, N. and Welty, C. (2002). Evaluating ontological decisions with ontoclean. *Communications of the ACM*, 45(2):61–65. See page 21.
- [Hamdy and Elsayed, 2018] Hamdy, A. and Elsayed, M. (2018). Automatic recommendation of software design patterns: Text retrieval approach. *J. Softw.*, 13(4):260–268. See pages 30, 32, and 33.
- [Hameed et al., 2002] Hameed, A., Sleeman, D. H., and Preece, A. D. (2002). Ontomanager: A workbench environment to facilitate ontology management and interoperability. In *EON*, pages 74–78. Citeseer. See page 21.
- [Heckmann et al., 2005] Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., and von Wilamowitz-Moellendorff, M. (2005). Gumo—the general user model ontology. In *International Conference on User Modeling*, pages 428–432. Springer. See page 60.
- [Henninger and Corrêa, 2007] Henninger, S. and Corrêa, V. (2007). Software pattern communities: Current practices and challenges. In *Proceedings of the 14th Conference on Pattern Languages of Programs*, pages 1–19. See page 51.
- [Hevner et al., 2004] Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pages 75–105. See page 6.
- [Hu and Liu, 2020] Hu, X. and Liu, J. (2020). Ontology construction and evaluation of uav fcms software requirement elicitation considering geographic environment factors. *IEEE Access*, 8:106165–106182. See page 21.
- [Hussain et al., 2018] Hussain, J., Ul Hassan, A., Muhammad Bilal, H. S., Ali, R., Afzal, M., Hussain, S., Bang, J., Banos, O., and Lee, S. (2018). Model-based adaptive user interface based on context and user experience evaluation. *Journal on Multimodal User Interfaces*, 12(1):1–16. See pages 34, 37, and 38.
- [Hussain et al., 2019] Hussain, S., Keung, J., Sohail, M. K., Khan, A. A., and Ilahi, M. (2019). Automated framework for classification and selection of software design patterns. *Applied Soft Computing*, 75:1–20. See pages 31, 32, and 33.
- [Jellad and Khemaja, 2014] Jellad, M. L. and Khemaja, M. (2014). Using an sws based integration approach for learning management systems adaptation and reconfiguration. In *2014 IEEE 23rd international WETICE conference*, pages 98–103. IEEE. See page 2.

- [Johannesson and Perjons, 2014] Johannesson, P. and Perjons, E. (2014). *An introduction to design science*, volume 10. Springer. See page 6.
- [Kalyanpur et al., 2006] Kalyanpur, A., Parsia, B., Sirin, E., Grau, B. C., and Hendler, J. (2006). Swoop: A web ontology editing browser. *Journal of Web Semantics*, 4(2):144–153. See page 19.
- [Kennard and Leaney, 2010] Kennard, R. and Leaney, J. (2010). Towards a general purpose architecture for ui generation. *Journal of Systems and Software*, 83(10):1896–1906. See page 2.
- [Khan and Khusro, 2020] Khan, I. and Khusro, S. (2020). Towards the design of context-aware adaptive user interfaces to minimize drivers’ distractions. *Mobile Information Systems*, 2020. See page 35.
- [Khan and Khusro, 2022] Khan, I. and Khusro, S. (2022). Context: context-aware adaptive sms client for drivers to reduce risky driving behaviors. *Soft Computing*, pages 1–18. See pages 35, 37, and 38.
- [Klyne, 2004] Klyne, G. (2004). Resource description framework (rdf): Concepts and abstract syntax. <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>. See page 15.
- [Kogut et al., 2002] Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., and Smith, J. (2002). Uml for ontology development. *The Knowledge Engineering Review*, 17(1):61–64. See page 19.
- [Kruschitz, 2009] Kruschitz, C. (2009). Xplml: a hci pattern formalizing and unifying approach. In *CHI’09 Extended Abstracts on Human Factors in Computing Systems*, pages 4117–4122. See page 14.
- [Kruschitz and Hitz, 2009] Kruschitz, C. and Hitz, M. (2009). The anatomy of hci design patterns. In *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, pages 202–207. IEEE. See page 12.
- [Kruschitz and Hitz, 2010] Kruschitz, C. and Hitz, M. (2010). Human-computer interaction design patterns: structure, methods, and tools. *Int. J. Adv. Softw*, 3(1). See page 13.
- [Kultsova et al., 2017] Kultsova, M., Potseluico, A., Zhukova, I., Skorikov, A., and Romanenko, R. (2017). A two-phase method of user interface adaptation for people with special needs. In *Conference on Creativity in Intelligent Technologies and Data Science*, pages 805–821. Springer. See page 60.
- [Kultsova et al., 2016] Kultsova, M., Romanenko, R., Anikin, A., and Pocuico, A. (2016). An ontology-based adaptation of user interface for people with special needs. In *Proceedings of the AINL FRUCT 2016 Conference*. See page 59.
- [Lantow, 2016] Lantow, B. (2016). Ontometrics: Putting metrics into use for ontology evaluation. In *KEOD*, pages 186–191. See page 21.
- [Letsu-Dake and Ntuen, 2009] Letsu-Dake, E. and Ntuen, C. A. (2009). A conceptual model for designing adaptive human–computer interfaces using the living systems theory. *Systems Research and Behavioral Science: The Official Journal of the International Federation for Systems Research*, 26(1):15–27. See page 2.
- [Li et al., 2015] Li, N., Hua, Q., Wang, S., Yu, K., and Wang, L. (2015). Research on a pattern-based user interface development method. In *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 1, pages 443–447. IEEE. See pages 28, 29.
- [Limbourg et al., 2004] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and Lopez-Jaquero, V. (2004). Usixml: A language supporting multi-path development of user interfaces. In *IFIP International Conference on Engineering for Human-Computer Interaction*, pages 200–220. Springer. See pages 3, 26, and 29.
- [López et al., 1999] López, M. F., Gómez-Pérez, A., Sierra, J. P., and Sierra, A. P. (1999). Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems and their applications*, 14(1):37–46. See pages 16, 17, and 52.

BIBLIOGRAPHY

- [Macik, 2012] Macik, M. (2012). Context model for ability-based automatic ui generation. In *2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 727–732. IEEE. See page 2.
- [Mahemoff and Johnston, 1998] Mahemoff, M. J. and Johnston, L. J. (1998). Pattern languages for usability: An investigation of alternative approaches. In *Proceedings. 3rd Asia Pacific Computer Human Interaction (Cat. No. 98EX110)*, pages 25–30. IEEE. See page 3.
- [March and Smith, 1995] March, S. T. and Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, 15(4):251–266. See page 6.
- [Mariño et al., 2018] Mariño, B. D. R., Rodríguez-Fórtiz, M. J., Torres, M. V. H., and Haddad, H. M. (2018). Accessibility and activity-centered design for ict users: Accesibilitic ontology. *IEEE Access*, 6:60655–60665. See page 60.
- [Märtin et al., 2017] Märtin, C., Herdin, C., and Engel, J. (2017). Model-based user-interface adaptation by exploiting situations, emotions and software patterns. In *CHIRA*, pages 50–59. See page 29.
- [Martin et al., 2002] Martin, D., Burstein, M., Denker, G., Hobbs, J., Kagal, L., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., et al. (2002). Daml-s: Semantic markup for web services. *DARPA Agent Markup Language*. See page 15.
- [Meixner et al., 2011] Meixner, G., Paternò, F., and Vanderdonckt, J. (2011). Past, present, and future of model-based user interface development. See page 26.
- [Miñón et al., 2016] Miñón, R., Paternò, F., Arrue, M., and Abascal, J. (2016). Integrating adaptation rules for people with special needs in model-based ui development process. *Universal Access in the Information Society*, 15(1):153–168. See pages 34, 37, and 38.
- [Moldovan et al., 2020] Moldovan, A., Nicula, V., Pasca, I., Popa, M., Namburu, J. K., Oros, A., and Brie, P. (2020). Openuidl, a user interface description language for runtime omni-channel user interfaces. *Proceedings of the ACM on Human-Computer Interaction*, 4(EICS):1–52. See pages 27, 29.
- [Molina et al., 2002] Molina, P. J., Meliá, S., and Pastor, O. (2002). Just-ui: A user interface specification model. In *Computer-Aided Design of User Interfaces III*, pages 63–74. Springer. See pages 27, 29.
- [Mu and Zeng, 2018] Mu, R. and Zeng, X. (2018). Collaborative filtering recommendation algorithm based on knowledge graph. *Mathematical Problems in Engineering*, 2018. See page 32.
- [Myers and Rosson, 1992] Myers, B. A. and Rosson, M. B. (1992). Survey on user interface programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 195–202. See page 2.
- [Naghdiipour and Hasheminejad, 2021] Naghdipour, A. and Hasheminejad, S. M. H. (2021). Ontology-based design pattern selection. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–7. IEEE. See pages 31, 32, and 33.
- [Neil, 2014] Neil, T. (2014). *Mobile design pattern gallery: UI patterns for smartphone apps*. " O'Reilly Media, Inc.". See page 59.
- [Nielsen, 1992] Nielsen, J. (1992). The usability engineering life cycle. *Computer*, 25(3):12–22. See page 129.
- [Nilsson, 2002] Nilsson, E. G. (2002). Combining compound conceptual user interface components with modelling patterns—a promising direction for model-based cross-platform user interface development. In *International Workshop on Design, Specification, and Verification of Interactive Systems*, pages 104–117. Springer. See pages 28, 29.
- [Nilsson, 2009] Nilsson, E. G. (2009). Design patterns for user interface for mobile applications. *Advances in engineering software*, 40(12):1318–1328. See page 59.

- [Norman et al., 1998] Norman, K. L., Shneiderman, B., Harper, B., and Slaughter, L. (1998). Questionnaire for user interaction satisfaction. *University of Maryland (Norman, 1989) Disponível em*. See page 129.
- [Noy et al., 2001] Noy, N. F., McGuinness, D. L., et al. (2001). Ontology development 101: A guide to creating your first ontology. See pages 16, 17.
- [Nunnally, 1994] Nunnally, J. C. (1994). *Psychometric theory 3E*. Tata McGraw-hill education. See page 122.
- [Pak and Zhou, 2009] Pak, J. and Zhou, L. (2009). A framework for ontology evaluation. In *Workshop on E-Business*, pages 10–18. Springer. See page 22.
- [Patel-Schneider, 2004] Patel-Schneider, P. F. (2004). Owl web ontology language semantics and abstract syntax, w3c recommendation. <http://www.w3.org/TR/2004/REC-owl-semantics-20040210/>. See pages 15, 20.
- [Paterno' et al., 2009] Paterno', F., Santoro, C., and Spano, L. D. (2009). Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(4):1–30. See pages 3, 27, and 29.
- [Peissner et al., 2012] Peissner, M., Häbe, D., Janssen, D., and Sellner, T. (2012). Myui: generating accessible user interfaces from multimodal design patterns. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 81–90. See pages 3, 34, 37, 38, and 59.
- [Petrasch, 2007] Petrasch, R. (2007). Model based user interface design: Model driven architecture und hci patterns. See page 2.
- [Pinto et al., 2004] Pinto, H. S., Staab, S., and Tempich, C. (2004). Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *ECAI*, volume 16, page 393. Citeseer. See pages 16, 17, and 52.
- [Planas et al., 2021] Planas, E., Daniel, G., Brambilla, M., and Cabot, J. (2021). Towards a model-driven approach for multiexperience ai-based user interfaces. *Software and Systems Modeling*, 20(4):997–1009. See pages 27, 29.
- [Porzel and Malaka, 2004] Porzel, R. and Malaka, R. (2004). A task-based approach for ontology evaluation. In *ECAI Workshop on Ontology Learning and Population, Valencia, Spain*, pages 1–6. Citeseer. See page 22.
- [Poveda-Villalón et al., 2014] Poveda-Villalón, M., Gómez-Pérez, A., and Suárez-Figueroa, M. C. (2014). Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34. See page 109.
- [Poveda-Villalón et al., 2012] Poveda-Villalón, M., Suárez-Figueroa, M. C., and Gómez-Pérez, A. (2012). Validating ontologies with oops! In *International conference on knowledge engineering and knowledge management*, pages 267–281. Springer. See page 21.
- [Powers, 2020] Powers, D. M. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*. See page 115.
- [Pu et al., 2011] Pu, P., Chen, L., and Hu, R. (2011). A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 157–164. See page 118.
- [Raad and Cruz, 2015] Raad, J. and Cruz, C. (2015). A survey on ontology evaluation methods. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. See page 21.

BIBLIOGRAPHY

- [Radeke et al., 2006] Radeke, F., Forbrig, P., Seffah, A., and Sinnig, D. (2006). Pim tool: Support for pattern-driven and model-based ui development. In *International Workshop on Task Models and Diagrams for User Interface Design*, pages 82–96. Springer. See pages [28](#), [29](#).
- [Rieger et al., 2020] Rieger, C., Lucrédio, D., Fortes, R. P. M., Kuchen, H., Dias, F., and Duarte, L. (2020). A model-driven approach to cross-platform development of accessible business apps. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 984–993. See pages [35](#), [37](#), and [38](#).
- [Rodrigues et al., 2011] Rodrigues, P. G., Raposo, A. B., and Soares, L. P. (2011). A virtual touch interaction device for immersive applications. *International Journal of Virtual Reality*, 10(4):1–10. See page [1](#).
- [Ruiz et al., 2019] Ruiz, J., Serral, E., and Snoeck, M. (2019). Evaluating user interface generation approaches: model-based versus model-driven development. *Software & Systems Modeling*, 18(4):2753–2776. See page [1](#).
- [Seffah, 2015a] Seffah, A. (2015a). Hci design patterns as a building block in model-driven engineering. In *Patterns of HCI Design and HCI Design of Patterns*, pages 35–58. Springer. See pages [12](#), [28](#), and [29](#).
- [Seffah, 2015b] Seffah, A. (2015b). *Patterns of HCI design and HCI design of patterns: bridging HCI design and model-driven software engineering*. Springer. See page [12](#).
- [Simon, 1996] Simon, H. A. (1996). The sciences of the artificial (vol. 136). See page [6](#).
- [Slimani, 2015] Slimani, T. (2015). Ontology development: A comparing study on tools, languages and formalisms. *Indian Journal of Science and Technology*, 8(24):1–12. See page [15](#).
- [Soui et al., 2017] Soui, M., Diab, S., Ouni, A., Essayeh, A., and Abed, M. (2017). An ontology-based approach for user interface adaptation. In *Advances in Intelligent Systems and Computing*, pages 199–215. Springer. See page [2](#).
- [SPSS, 2013] SPSS, I. (2013). *Ibm spss statistics for windows*. Armonk, New York, USA: IBM SPSS, 2. See pages [119](#), [130](#).
- [Studer et al., 1998] Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1-2):161–197. See page [15](#).
- [Suárez-Figueroa et al., 2012] Suárez-Figueroa, M. C., Gómez-Pérez, A., and Fernández-López, M. (2012). The neon methodology for ontology engineering. In *Ontology engineering in a networked world*, pages 9–34. Springer. See pages [16](#), [18](#), [52](#), [55](#), and [105](#).
- [Sunyé et al., 2000] Sunyé, G., Guennec, A. L., and Jézéquel, J.-M. (2000). Design patterns application in uml. In *European Conference on Object-Oriented Programming*, pages 44–62. Springer. See page [14](#).
- [Sure et al., 2002] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D. (2002). Ontoedit: Collaborative ontology development for the semantic web. In *International semantic web conference*, pages 221–235. Springer. See page [19](#).
- [Sure et al., 2004] Sure, Y., Staab, S., and Studer, R. (2004). On-to-knowledge methodology (otkm). In *Handbook on ontologies*, pages 117–132. Springer. See pages [16](#), [18](#), and [52](#).
- [Thanh-Diane et al., 2016] Thanh-Diane, N., Vanderdonckt, J., and Seffah, A. (2016). Uiplml: pattern-based engineering of user interfaces of multi-platform systems. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12. IEEE. See page [14](#).
- [Tidwell, 1999] Tidwell, J. (1999). Common ground: A pattern language for human-computer interface design. See page [13](#).
- [Tidwell, 2010] Tidwell, J. (2010). *Designing interfaces: Patterns for effective interaction design*. "O'Reilly Media, Inc.". See page [59](#).

-
- [Vanderdonckt and Simarro, 2010] Vanderdonckt, J. and Simarro, F. M. (2010). Generative pattern-based design of user interfaces. In *Proceedings of the 1st international workshop on pattern-driven engineering of interactive computing systems*, pages 12–19. See page 28.
- [Vora, 2009] Vora, P. (2009). *Web application design patterns*. Morgan Kaufmann. See page 13.
- [Vrandečić, 2009] Vrandečić, D. (2009). Ontology evaluation. In *Handbook on ontologies*, pages 293–313. Springer. See page 20.
- [Wang et al., 2018] Wang, J., Cao, B., Yu, P., Sun, L., Bao, W., and Zhu, X. (2018). Deep learning towards mobile applications. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1385–1393. IEEE. See page 1.
- [Welie, 1999] Welie, M. (1999). Patterns in interaction design: the amsterdam collection. See page 13.
- [Wetchakorn and Prompoon, 2015] Wetchakorn, T. and Prompoon, N. (2015). Method for mobile user interface design patterns creation for ios platform. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 150–155. IEEE. See page 59.
- [Wieringa, 2009] Wieringa, R. (2009). Design science as nested problem solving. In *Proceedings of the 4th international conference on design science research in information systems and technology*, pages 1–12. See page 6.
- [Wieringa, 2014] Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer. See page 6.
- [Yigitbas et al., 2020] Yigitbas, E., Jovanovikj, I., Biermeier, K., Sauer, S., and Engels, G. (2020). Integrated model-driven development of self-adaptive user interfaces. *Software and Systems Modeling*, 19(5):1057–1081. See pages 35, 37, and 38.
- [Youssef et al., 2020] Youssef, C. K., Ahmed, F. M., Hashem, H. M., Talaat, V. E., Shorim, N., and Ghanim, T. (2020). Gqm-based tree model for automatic recommendation of design pattern category. In *Proceedings of the 2020 9th International Conference on Software and Information Engineering (ICSIE)*, pages 126–130. See pages 31, 32, and 33.
- [Zavitsanos et al., 2010] Zavitsanos, E., Paliouras, G., and Vouros, G. A. (2010). Gold standard evaluation of ontology learning methods through ontology transformation and alignment. *IEEE Transactions on knowledge and data engineering*, 23(11):1635–1648. See page 22.