

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Murillo Vasconcelos Henriques Bittencourt Castro

**An Ontology to support Knowledge
Management Solutions for Human-
Computer Interaction Design**

VITÓRIA
2021

Murillo Vasconcelos Henriques Bittencourt Castro

An Ontology to support Knowledge Management Solutions for Human- Computer Interaction Design

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Orientador(a): Monalessa Perini Barcellos

Coorientador: Ricardo de Almeida Falbo
(in memoriam)

VITÓRIA
2021

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

C355a Castro, Murillo Vasconcelos Henriques Bittencourt, 1991-
An Ontology to support Knowledge Management Solutions for Human-Computer Interaction Design / Murillo Vasconcelos Henriques Bittencourt Castro. - 2021.
162 f. : il.

Orientadora: Monalessa Perini Barcellos.
Coorientador: Ricardo de Almeida Falbo.
Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Gestão do conhecimento. 2. Engenharia de software. 3. Interação homem-máquina. I. Barcellos, Monalessa Perini. II. Falbo, Ricardo de Almeida. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Murillo Vasconcelos Henriques Bittencourt Castro

An Ontology to support Knowledge Management Solutions for Human- Computer Interaction Design

COMISSÃO EXAMINADORA

Prof. Monalessa Perini Barcellos, D. Sc.
Universidade Federal do Espírito Santo (UFES)
Orientadora

Prof. Vítor Estêvão Silva Souza, Ph. D.
Universidade Federal do Espírito Santo (UFES)
Avaliador Interno

Profa. Tayana Uchôa Conte, D. Sc.
Universidade Federal do Amazonas (UFAM)
Avaliador Externo

Vitória, 19 de maio de 2021

This work is dedicated to my parents, Estêvão and Eliana, in return for their investment in my education, and to the memory and the legacy of Prof. Ricardo Falbo.

ACKNOWLEDGEMENTS

When this work began, I had absolutely no idea what the result would be. Since then, many things have happened, causing substantial changes in me and in the world. Today, three years and two months later, the balance is undoubtedly positive, despite the losses and challenges along the way. From this experience, the main learnings I will take to my life are to remain strong and comfortable in the face of discomfort, to enjoy the process regardless of the result, and to celebrate the achievements, like the completion of this work. Celebrating is also an act of thanking those who participated in the journey. Here is my gratitude to everyone who shared moments with me during this period, especially those who participated and contributed directly to this achievement.

I am immeasurably grateful to my Father and Mother for giving me my life, for nurturing me with great love, and for being the main and best personal references I could have. To my Sisters, my nephew, my niece, and other familiars and friends, for the moments of fun, affection, and support. To Hosana Dias, and to Hilario Trigo & Escola Vida, for their help in providing me psychological resources that enable me to live in peace with my thoughts and emotions.

I have no words to thank Prof. Monalessa and Prof. Falbo, for having guided me along this journey and for inspiring me as great models of teachers, supervisors, and people. The synergy between them is so great that it is pretty hard to mention how much each of them contributed separately to me and to this work. I particularly admire Monalessa's strength for moving forward after Falbo having an unfortunate and early passing, and I'm sure he is proud of our work, wherever he is. After all, his presence remains and will remain alive through his legacy and us.

I am also grateful to the other NEMO senior members, namely João Paulo, Giancarlo, Renata, and Vítor, for the knowledge transmitted and inspiration, mainly during their classes; to Simone and Bia for the partnership and collaboration in our works; to the members of the examination board for their availability to evaluate this work; to Resultate, for having been the environment that provided the early motivation for this work; and to the participants of the studies, for their time availability to contribute to this work.

Finally, I am grateful to PPGI, UFES, and the Federative Republic of Brazil, for offering this postgraduate program that contributed to my academic, professional, and personal growth.

ABSTRACT

Developing interactive systems is a challenging task. It involves concerns related to the human-computer interaction (HCI), such as usability and user experience. Therefore, HCI design must be addressed when developing such systems. HCI design often involves people with different backgrounds, technical languages, terms and knowledge, what makes communication and knowledge transfer a challenging issue. In this scenario, knowledge management can support understanding concepts from different knowledge areas and help learn from previous experiences. Knowledge management has supported HCI design mainly to improve product quality and reduce effort and time spent on design activities. However, there is a need for simpler and more practical knowledge management solutions to support HCI design. In addition, the lack of a common conceptualization about HCI design has been one of the main challenges to be addressed. This leads to semantic interoperability problems, such as ambiguity and imprecision when interpreting shared information, and hampers communication and knowledge transfer. Aiming to provide a well-founded conceptualization about HCI design domain in the context of the development of interactive systems, this work proposes HCIDO (Human-Computer Interaction Design Ontology). HCIDO is a reference ontology of the Human-Computer Interaction Ontology Network (HCI-ON) and is also connected to the Software Engineering Ontology Network (SEON), allowing for the reuse of concepts related to Software Engineering and HCI aspects, such as requirements, code, interactive systems and users, as well as making them connected to design aspects. HCIDO was evaluated through verification and validation techniques. Moreover, a computational tool was developed using HCIDO as a reference model, illustrating how the ontology can be applied to support knowledge management solutions in HCI design. The tool supports knowledge management activities (e.g., knowledge capture, representation, storage, retrieval, use and evaluation) in the HCI design of interactive systems by allowing HCI designers to annotate structured information about design choices in design artifacts shared with HCI design stakeholders.

Keywords: HCI Design, Knowledge Management, Ontology

TABLE OF CONTENTS

Chapter 1 – Introduction	9
1.1 Context	9
1.2 Motivation.....	11
1.3 Objectives	12
1.4 Research Approach	13
1.5 Structure of this Document	17
Chapter 2 – Background	18
2.1 HCI Design.....	18
2.2 Knowledge Management.....	20
2.3 Ontologies.....	24
2.3.1 UFO (Unified Foundational Ontology).....	26
2.3.2 SEON (<i>Software Engineering Ontology Network</i>)	28
2.3.3 HCI-ON (<i>Human-Computer Interaction Ontology Network</i>)	31
2.4 Ontologies in HCI Design context.....	35
2.5 Concluding Remarks	36
Chapter 3 – Systematic Mapping: KM in HCI Design	37
3.1 Introduction.....	37
3.2 Research protocol	38
3.3 Results.....	40
3.4 Discussion.....	48
3.5 Threats to validity	51
3.6 Concluding Remarks	53
Chapter 4 – Survey: KM in HCI Design Practice	54
4.1 Introduction.....	54
4.2 Survey Planning and Execution.....	55
4.3 Results.....	57
4.4 Discussion.....	63
4.5 Threats to Validity	66
4.6 Consolidated View of Findings	67
4.7 Concluding Remarks	69
Chapter 5 – Human-Computer Interaction Design Ontology.....	71
5.1 Introduction.....	71

5.2	Software Design Reference Ontology	73
5.2.1	SDRO Evaluation.....	80
5.2.2	Discussion.....	86
5.3	Human-Computer Interaction Design Ontology.....	87
5.3.1	HCIDO Evaluation.....	95
5.3.2	Discussion.....	103
5.4	Related Works	104
5.5	Concluding Remarks	105
Chapter 6 – KTID: A Computational Tool to Support KM Aspects in HCI Design		106
6.1	Introduction.....	106
6.2	KTID: <i>Knowledge Tool for Interaction Design</i>	107
6.3	Evaluating KTID.....	116
6.3.1	Study Planning.....	116
6.3.2	Study Execution.....	118
6.3.3	Study Results	119
6.3.4	Discussion.....	121
6.3.5	Threats to Validity	123
6.4	Concluding Remarks	124
Chapter 7 – Final Considerations and Future Work		126
7.1	Final Considerations.....	126
7.2	Contributions.....	128
7.3	Future Work	128
References		131
Appendix A – Artifacts used in KTID Evaluation Study		143
A.1	Instructions Document.....	143
A.2	Consent Form	153
A.3	Participants Profile Form	154
A.4	Feedback Questionnaire	158

Chapter 1

Introduction

This chapter presents the context, motivation and objectives of this work, as well as the adopted research approach and the structure of this dissertation.

1.1 Context

The interest in interactive systems and their impact on people's life has promoted the study and practice of usability (CARROLL, 2014). Usability is a key aspect to a successful interactive system and is related to user efficiency and satisfaction when interacting with the system. To an interactive system reach high usability levels, it is necessary to take human-computer interaction (HCI) design aspects into account during its development process (CARROLL, 2014).

HCI is concerned with usability and other aspects related to the interaction between users and computer systems, necessary to produce more usable software (CARROLL, 2014). It involves knowledge from multiple fields, such as ergonomics, cognitive science, user experience, human factors, among others (SUTCLIFFE, 2014). Due to the diverse body of knowledge involved when designing HCI aspects of interactive systems, interactive system development teams are frequently multidisciplinary, joining people from different backgrounds, with their own technical language, terms and knowledge. Even the conceptualization about the product may be conflicting among different stakeholders, which hampers communication and knowledge transfer (CARROLL, 2014; ROGERS; SHARP; PREECE, 2011).

Developing interactive systems is a knowledge-intensive task. Knowledge Management (KM) principles and practices have been successfully applied to support knowledge capture, storage, use and transfer in the software development context in general (RUS; LINDVALL, 2002; VALASKI; MALUCELLI; REINEHR, 2012). KM can also be helpful to address challenges in the HCI design of interactive systems, since it might provide support to capture and represent knowledge in an accessible and reusable way. For example, design solutions developed by an organization can be stored and related to the requirements that motivate them, components and patterns used to build them and evaluation results. As a result, the team can learn from previous experiences and share a common understanding

about the system, contributing to produce better products and perform processes more efficiently.

HCI is a wide domain and as the area is getting more mature, new terms are proposed and new meanings are assigned to existing terms. Consequently, it is not trivial to have a common conceptualization of HCI, leading to semantic interoperability problems, such as ambiguity and imprecision when interpreting shared information. Moreover, the integration of HCI knowledge and practices into Software Engineering (SE) processes involves additional challenges due to the knowledge intersection between them (OGUNYEMI; LAMAS, 2014). For example, a lot of HCI related terms are also related to SE, such as system, requirement, design and user interface. However, since HCI is more user centered (i.e., more concerned on the tasks that users can perform with an interactive system to achieve their goals) and SE is more system centered (i.e., more concerned on the functions that an interactive system must provide to satisfy its requirements), different meanings can be associated to the same term depending on the context (e.g., designers may refer to the user interface as what they see through the graphical elements displayed on the screen, while developers may refer to it as the portion of the code which produces the output displayed in the screen) (OGUNYEMI; LAMAS, 2014).

Ontologies can be used to capture and organize knowledge based on a common vocabulary to deal with interoperability and knowledge-related problems. An ontology is a formal and explicit specification of a shared conceptualization (STUDER; BENJAMINS; FENSEL, 1998). In the HCI context, they have been applied to knowledge representation, to aid in interaction design and evaluation, interface adaptation, semantic annotation, among others (COSTA *et al.*, 2020). For a complex domain, representing its knowledge as a single ontology results in a large and monolithic ontology that is hard to manipulate, use, and maintain (SUÁREZ-FIGUEROA *et al.*, 2012). On the other hand, representing each sub-domain in isolation is a costly task that leads to a very fragmented solution that is again hard to handle (RUY *et al.*, 2016). In such cases, building an ontology network is an adequate solution. An ontology network (ON) is a collection of ontologies related together by means of dependency and alignment relationships (SUÁREZ-FIGUEROA *et al.*, 2012). Being a complex and wide domain, HCI ontologies should be organized in an ON. Moreover, considering the strong relation between HCI and SE, HCI ontologies should reuse concepts from SE ontologies. Hence, the development of an ontology about HCI design, integrated into an HCI ontology network and reusing concepts from SE ontologies, may be useful to support knowledge management solutions in the HCI design of interactive systems.

1.2 Motivation

HCI has received more and more attention in the software development context. It is concerned with *“the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them”* (HEWETT *et al.*, 1992). HCI addresses important aspects of a successful interactive system, such as usability, user experience, and accessibility (CARROLL, 2014). Hence, there have been efforts to integrate HCI knowledge and practices into Software Engineering processes, and there is still a lot of work to be done in this context (HARNING; VANDERDONCKT; FLORINS, 2003; SEFFAH; GULLIKSEN; DESMARAIS, 2005).

As any general design activity, HCI design of interactive systems embodies a large amount of tacit knowledge, which cannot be easily articulated (BOFYLATOS; SPYROU, 2017). The lack of mechanisms to make tacit knowledge explicit leads to communication and knowledge transfer challenges in HCI design. For example, a designer may not be able to point out the reasons why some design choices were made and describe them in artifacts. As a result, other stakeholders (e.g., developers and project managers) may have a wrong or incomplete understanding about the design and, thus, other designers may not be able to reuse the solution in future designs. Therefore, it is important to effectively manage HCI design knowledge in interactive systems development.

Knowledge Management (KM) principles and methods can be helpful to address the large amount of tacit knowledge involved in HCI design, since they aim to transform tacit and individual knowledge into explicit and shared knowledge. By raising individual knowledge to the organizational level, KM promotes knowledge propagation and learning, making knowledge accessible and reusable across the entire organization (O'LEARY, 1998; RUS; LINDVALL, 2002; SCHNEIDER, 2009). KM solutions (e.g., knowledge management systems and knowledge-based systems) can be supported by ontologies to provide knowledge access, optimize knowledge retrieval, support communication mechanisms and, therefore, knowledge exchange (VARMA, 2007). Thus, the use of ontologies combined with KM solutions in HCI design can help enhance reuse and facilitate reasoning and inferences on existing HCI design knowledge.

Since HCI and SE are strongly related, it is important to consider aspects from both domains when developing ontologies related to HCI design. Considering that SE and HCI are complex and wide domains, ontologies addressing these domains should be organized as ontology networks. ONs can be used to establish a comprehensive conceptualization that

provides a common understanding about the domain and can be used as a reference to solve semantic interoperability and knowledge problems related to the conceptualization as a whole or to extracts of it. In this sense, Ruy *et al.* (2016) proposed SEON, the Software Engineering Ontology Network, which contains ontologies addressing several SE subdomains and forming a network with a comprehensive and consistent conceptualization of SE. In an analogous initiative, to address the HCI domain, Costa *et al.* (2020) have developed HCI-ON, the Human-Computer Interaction Ontology Network, aiming to establish a comprehensive conceptualization of HCI by including ontologies that provide knowledge to talk about the whole life cycle of an HCI project (e.g., design, UI, modalities of interaction, evaluation and context of use) (COSTA *et al.*, 2020). Thus, the inclusion of an ontology about HCI design in HCI-ON is required to address what is referred as HCI Engineering (HEFLEY *et al.*, 1994), providing knowledge to talk about important aspects of the development of interactive systems, such as how a designer designs the user interface of an interactive system and what is the relation between the design, the requirements and the actual system that users interact with.

In view of the above, this work explores the combination of KM foundations with ontologies and ontology networks to potentialize knowledge management solutions (i.e., solutions, automated or not, that support knowledge management activities such as knowledge capture, representation, storage, retrieval, use or assessment) in the context of the HCI design of interactive systems.

1.3 Objectives

This work has the **main objective** of *proposing a well-founded consensual conceptualization of HCI design to support knowledge management solutions to aid in HCI design of interactive systems*. This main objective can be detailed in the following **specific objectives**:

- (i) Investigate the state of the art about knowledge management in HCI design: this goal is intended to investigate how knowledge management has been used to support HCI design and identify gaps that have not been addressed by the proposed knowledge management solutions;
- (ii) Investigate the state of the practice about knowledge management in HCI design: aims at investigating knowledge management aspects (practices, technologies, artifacts, etc.) in HCI design practice;

- (iii) Develop a reference ontology about HCI design of interactive systems: aims at building a reference ontology to provide a well-founded consensual conceptualization of HCI design;
- (iv) Apply the reference ontology to support HCI design of interactive systems: intends to use the reference ontology in the development of a computational tool to support knowledge management aspects in HCI design.

1.4 Research Approach

The research method adopted in this work followed the Design Science Research (DSR) paradigm, which concerns extending human and organizational capabilities by creating new and innovative artifacts (HEVNER, 2007). It comprises the following steps (PEFFERS *et al.*, 2007): (i) *Problem identification and motivation*; (ii) *Define the objectives for a solution*; (iii) *Design and development*; (iv) *Demonstration*; (v) *Evaluation*, and (vi) *Communication*. These six steps are associated with three cycles that characterize DSR as an iterative process, as defined by Hevner (2007): *Relevance Cycle*, *Design Cycle* and *Rigor Cycle*. The *Relevance Cycle* involves defining the problem to be addressed, the research requirements, and the criteria to evaluate research results, including steps (i) and (ii). The *Design Cycle* involves developing and evaluating artifacts or theories to solve the identified problem, comprising steps (iii), (iv) and (v). Finally, the *Rigor Cycle* refers to using and generating knowledge, consisting in step (vi) plus the use of knowledge and foundations along the work.

In the “*Problem identification and motivation*” step, the problem was first identified in practice by the author of this dissertation, when working on the development of interactive systems as a software engineer together with HCI designers. This author noticed problems to share knowledge about HCI design decisions and difficulties to achieve a harmonized view of the system from HCI designers’ and developers’ perspectives. Since they had different views of the interactive system and different understandings of HCI design and its relation to other aspects of software development, it was hard to establish a consensual communication protocol and reuse knowledge from developed HCI design solutions. Thus, an informal literature review was performed to learn about the research topic. As a result, the problem to be focused by this work was established as the need to address difficulties involved in managing knowledge in the HCI design of interactive systems. Aiming to understand the subject in deep, we investigated the state of the art about knowledge management in HCI design through a systematic mapping. The mapping results indicated, among other results, that (i) the lack of a common conceptualization about HCI design leads

to communication problems between the different actors involved in the HCI design process; and (ii) it is necessary to take knowledge management solutions into practical HCI design environments to reduce the gap between theory and practice. After that, to complement the mapping results and give us a better understanding about knowledge management in HCI design in practice, we carried out a survey with 39 HCI design professionals. The survey results reinforced the lack of a consensual conceptualization about HCI design as a challenge and indicated that professionals have been concerned in managing HCI design knowledge, preferring informal and simpler methods and tools.

Considering the identified problem, the gaps pointed by the mapping, the survey results, the benefits reported in the literature of using ontologies to address semantic interoperability problems and the potential of ontologies to contribute with knowledge management solutions, in the step “*Define the objectives for a solution*” we decided to develop a reference ontology about HCI design of interactive systems. As requirements to the reference ontology, we defined: (R1) the ontology must cover main aspects regarding HCI design, including not only the created artifacts but also mental aspects that precede the creation of design artifacts (e.g., the choices made by the designer regarding which elements will be used); (R2) the ontology must consider aspects related from both HCI and SE; (R3) the ontology must be modular; (R4) the ontology must be formally rigorous; (R5) the ontology must be grounded in a well-founded ontology; (R6) the ontology must be developed by following an appropriate Ontology Engineering method; and (R7) the ontology must be used to solve problems. These requirements were established based on some characteristics of “beautiful ontologies”. A beautiful ontology is one that reflects an elegant solution for modeling a problem and it is at the same time good (in terms of formal quality), usable and practicable (D’AQUIN; GANGEMI, 2011). In addition to the requirements to be met by the ontology, based on (FALBO, 2014), we defined the following criteria to evaluate it: (C1) the ontology elements (i.e., concepts, relations and axioms) must be the ones sufficient and necessary to cover the scope defined by means of competency questions; and (C2) the ontology must be able to represent real-world situations. Moreover, to evaluate the ontology use (i.e., R7), we defined that (C3) the solution built based on the ontology must be feasible and useful.

During the “*Design and development*” step we developed the HCI Design Ontology (HCIDO), the main artifact proposed in this work. To address R1, HCIDO is based on HCI design literature, standards and also in theories related to design in general. To meet R2, HCIDO was developed as a networked ontology of HCI-ON (COSTA *et al.*, 2020) and

reuses concepts from SEON ontologies (RUY *et al.*, 2016), particularly from the Software Design Reference Ontology (SDRO), which was also developed in the context of this work and reuses concepts from other SEON ontologies, namely: Software Process Ontology (SPO) (BRINGUENTE; FALBO; GUIZZARDI, 2011); System and Software Ontology (SysSwO) (BRINGUENTE; FALBO; GUIZZARDI, 2011; DUARTE *et al.*, 2018); and Software Requirements Ontology (RSRO) (DUARTE *et al.*, 2018). To satisfy R3, HCIDO is organized into two sub-ontologies. To meet R4, we defined HCIDO by means of conceptual models (represented in UML - Unified Modeling Language) and textual descriptions. Concerning R5, we grounded HCIO in UFO (GUIZZARDI, 2005). As for R6, we followed SABiO (Systematic Approach for Building Ontologies) (FALBO, 2014). Then, in the “*Demonstration*” step, we used HCIDO as a reference model in the development of the Knowledge Tool for Interaction Design (KTID), a computational tool to support HCI design knowledge management aspects in the development of interactive systems. During the *Evaluation* step, to evaluate HCIDO considering C1 and C2, we performed verification and validation activities, as suggested in SABiO (FALBO, 2014). To evaluate HCIDO considering C3, we performed a study in which two HCI designers used KTID in an HCI design scenario.

Finally, the “*Communication*” step involves presenting the research results to the Academic and Industry communities, which involved elaborating this dissertation and some papers (CASTRO *et al.*, 2020; COSTA *et al.*, 2020) published in the context of this research. The main contribution of this work is HCIDO, a reference ontology providing a well-founded conceptualization about HCI design. There are also other contributions: (i) the systematic mapping investigating knowledge management in HCI design; (ii) the survey investigating knowledge management aspects in HCI design practice; (iii) SDRO, a reference ontology about design in the software context; and (iv) KTID, a computational tool based on HCIDO to support knowledge management aspects in the HCI design of interactive systems. Figure 1.1 summarizes the Design Science cycles performed in this work.

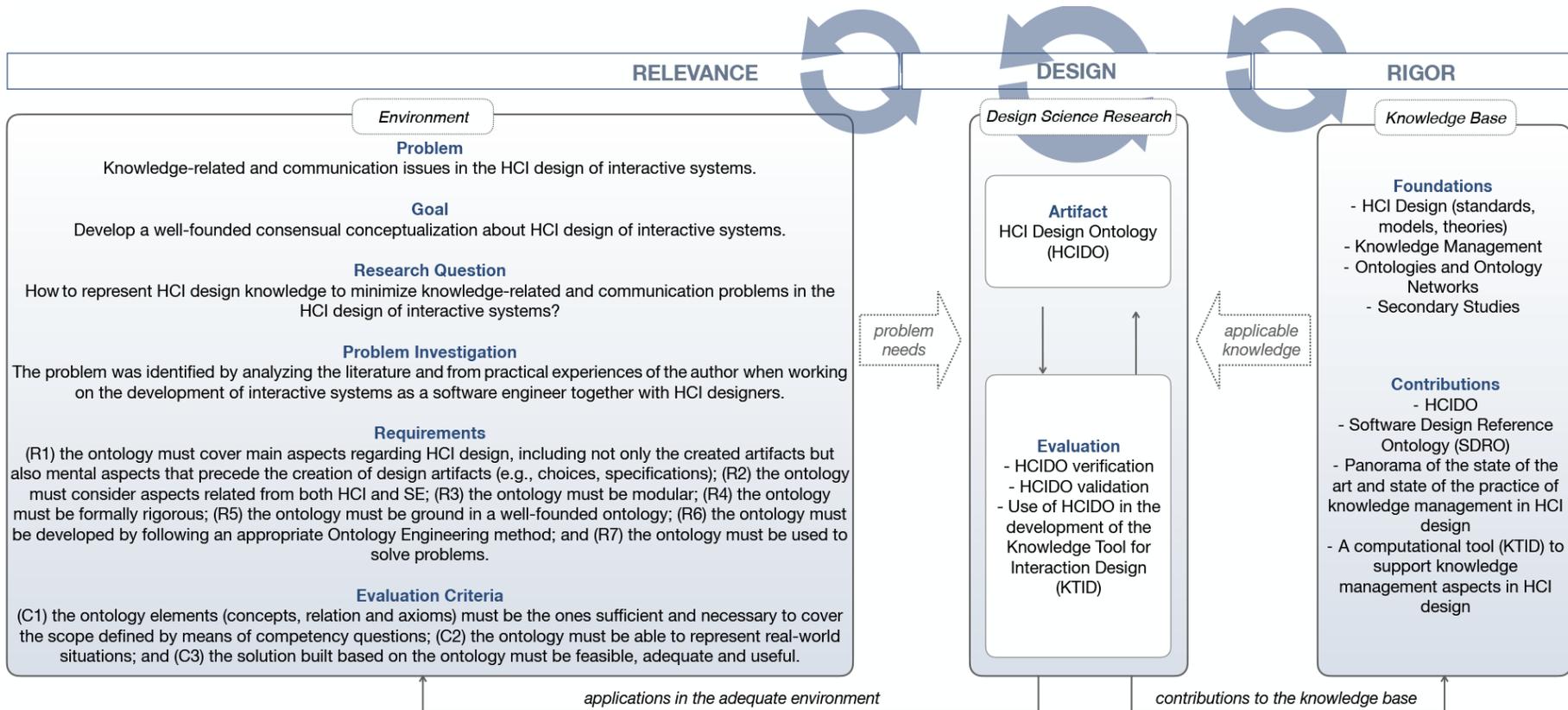


Figure 1.1 – Overview of Design Science cycles applied in this work.

1.5 Structure of this Document

This initial chapter presented the main ideas about this dissertation, describing the context, motivations, objectives and followed research approach. Besides this introduction, this dissertation is composed of the following chapters and appendices:

- **Chapter 2 (Background):** presents the background for the work, including content related to HCI design, knowledge management, ontologies and ontologies in HCI design.
- **Chapter 3 (Systematic Mapping: KM in HCI Design):** presents a systematic mapping that investigated the use of knowledge management in HCI design according to the literature.
- **Chapter 4 (Survey: KM in HCI Design Practice):** presents a survey carried out with HCI design practitioners aiming to investigate knowledge management in HCI design practice.
- **Chapter 5 (Human-Computer Interaction Design Ontology):** presents HCIDO, the reference ontology about HCI design proposed in this work. For that, the chapter also presents Software Design Reference Ontology (SDRO), which was developed in this work to be reused in HCIDO development.
- **Chapter 6 (KTID: A Computational Tool to Support KM Aspects in HCI Design):** presents KTID, the computational tool developed based on HCIDO to support knowledge management aspects in the HCI design of interactive systems.
- **Chapter 7 (Final Considerations and Future Work):** presents our final considerations, contributions and proposals of future works to continue and improve the work proposed in this dissertation.
- **Appendix A (Artifacts used in KTID Evaluation Study):** presents the forms and the instructions document used in KTID evaluation study.

Chapter 2

Background

This chapter presents the background for this work. Section 2.1 addresses HCI Design. Section 2.2 concerns Ontologies. Section 2.3 regards Ontologies in the HCI design context. Section 2.4 addresses Knowledge Management. Finally, Section 2.5 presents the chapter concluding remarks.

2.1 HCI Design

Human-Computer Interaction (HCI) can be defined as the discipline responsible for the analysis, design, implementation and evaluation of interactive computer systems for human use (ROGERS; SHARP; PREECE, 2011). This discipline has evolved since the 1980s through various terminologies, classifications and studies. An important study, for instance, defines three paradigms to explain the HCI phenomenon (HARRISON; TATAR; SENEGERS, 2007). The first paradigm sees interaction as man-machine coupling, aims at optimizing fit between man and machine, and mixtures engineering and human factors. The second focuses on cognitive science and adopts the metaphor of mind and computer as coupled information processors and aims at optimizing accuracy and efficiency of information transfer. The third sees interaction as phenomenologically situated, has in its center the meaning and meaning construction, and aims at supporting situated actions in the world.

An interactive computer system (also referred in this work as “interactive system”) is a combination of hardware and software that receives input from and communicates output to users (ISO, 2019). Dix *et al.* (2003) consider the communication between users and interactive computer systems the interaction itself. User and system are, thus, participants in the interaction. Briefly, a human-computer interaction is the communication process that occurs during the use of an interactive computer system and that involves user actions on the system interface (user input) and user interpretations of the system responses (system output) revealed through the user interface (Figure 2.1). The user interface includes all parts of the system that a user has contact with, physically, perceptually or conceptually (BENYON, 2013). Interactive computer systems aid in goals achievement by supporting the accomplishment of tasks in some application domain or context of use where users interact with the system through its interface.

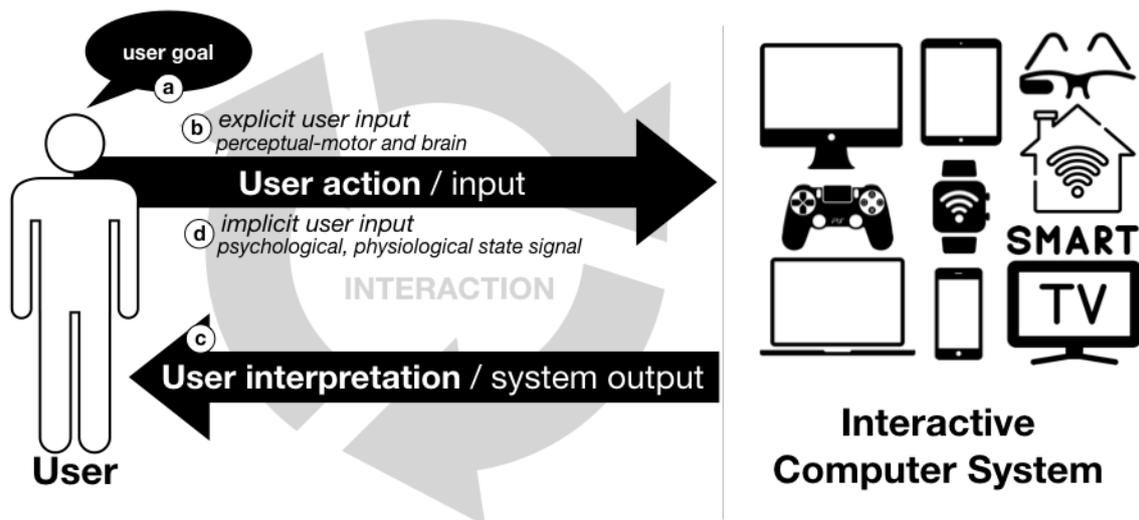


Figure 2.1 – Human-Computer Interaction: (a) user goal triggering the interaction, (b) user action and explicit user input, (c) system output (triggering the interaction or not) and user interpretation, (d) user action (user internal state) and implicit user input (COSTA, 2021).

According to Norman (2013), the interaction cycle can start from the top, in a goal-driven behavior (Figure 2.1, (a)), where the user first establishes a goal to be achieved and then goes through user actions to accomplish the goal. In Figure 2.1, (a) together with (b) represents that the interaction starts with the goal establishment and a user action that triggers the interaction cycle. The interaction cycle can also start from the bottom, in a data-driven or event-driven behavior, triggered by some event in the world (e.g., an event caused by an interactive computer system) and then can go through user actions (Figure 2.1, (c) when the system output triggers the interaction). This perspective refers to an asymmetrical interaction, which has been the most predominant HCI mode (KROL *et al.*, 2016). In asymmetrical interaction, the system receives (explicit) user input through user perceptual-motor and brain (GALLAGHER, 2006). Contemporary advances in HCI have led to symmetrical interaction, where the system captures (implicit) user input through psychological and physiological state (Figure 2.1, (d)) (FAIRCLOUGH, 2009). In both cases, inputs can result from intentional or unintentional actions of the user. Symmetrical interaction and asymmetrical interaction can occur concomitantly (COSTA, 2021).

HCI design focuses on how to design the interactive computer system to support users to achieve their goals through the interaction between them and the system (SUTCLIFFE, 2014). It is concerned with usability and other important attributes such as user experience, accessibility and communicability. Usability is the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use (ISO, 2019). It addresses the effort

and ease of the user during the interaction, considering her cognitive, perceptive and motor skills. According to Rogers, Sharp and Preece (2011), user experience relates to users' emotions and feelings and is essential for interaction design because it takes into account how a product behaves and is used by people in the real world. Accessibility refers to the removal of barriers that prevent interface and interaction access. Finally, communicability concerns the ability of the interface to communicate design logic to the user (DE SOUZA, 2005).

HCI design is user-centered, hence it is said User-Centered Design (UCD) (CHAMMAS; QUARESMA; MONT'ALVÃO, 2015). UCD is based on ergonomics, usability and human factors. It focuses on the use and development of interactive systems, with emphasis on making products usable and understandable. It puts human needs, capabilities and behavior first, then designs the system to accommodate them. Its main principles are user focus (its characteristics, needs and objectives), observable metrics (user performance and reactions) and iterative design (repeat as often as needed) (CHAMMAS; QUARESMA; MONT'ALVÃO, 2015; ISO, 2019). The term Human-Centered Design (HCD) has been adopted in place of UCD to emphasize the impact on all stakeholders and not just on those considered users (ISO, 2019).

In general, the HCI design process comprises four main activities: *understand and specify context of use*, which aims to study the product users and intended uses; *specify requirements*, which aims to identify user needs and specify functional and other requirements for the product; *produce design solutions*, which aims to achieve the best user experience and includes the production of artifacts such as prototypes and mockups that will be used in the future as a basis for developing the system; and *evaluation*, when the user evaluates the results produced in the previous activities (ISO, 2019).

2.2 Knowledge Management

Schneider (2009) defines knowledge as a human specialty stored in people's minds, acquired through experience and interaction with their environment. Knowledge helps software organizations to react faster and better, supporting more accurate and precise responses, which contributes to increase software quality and client satisfaction (SCHNEIDER, 2009).

According to Polanyi (1966), knowledge can be classified in two types: tacit knowledge and explicit knowledge. *Tacit knowledge* represents the subjective and non-documented knowledge that lies in people's mind. This kind of knowledge is related to

personal experience and involves intangible factors such as beliefs, perspectives, intuition and values. *Explicit knowledge*, in turn, represents objective knowledge that can be documented in such a way that it can be accessed by other people. Knowledge in this format can be easily transmitted and shared in the form of general principles, scientific formulas, codified procedures, among others.

The creation of knowledge in an organization is considered by Nonaka (1994) as a dynamic and continuous interaction between tacit and explicit knowledge. This interaction (outlined in Figure 2.2) can happen in four different modes of knowledge conversion, namely:

- i. **Socialization:** is the interchange and creation of tacit knowledge through interaction between individuals. In this mode, tacit knowledge can be acquired by an individual without using language, by means of observation, imitation and practice.
- ii. **Internalization:** is the process in which explicit knowledge is transformed into tacit knowledge, usually realized by reading documents, for example.
- iii. **Externalization:** is the transformation of tacit knowledge into explicit knowledge through the symbolic representation of the tacit knowledge, usually in the form of models, descriptions, sketches, among others.
- iv. **Combination:** is the exchange of explicit knowledge that creates new explicit knowledge, for example, two different spreadsheets joined to provide new knowledge through the combination of their data.

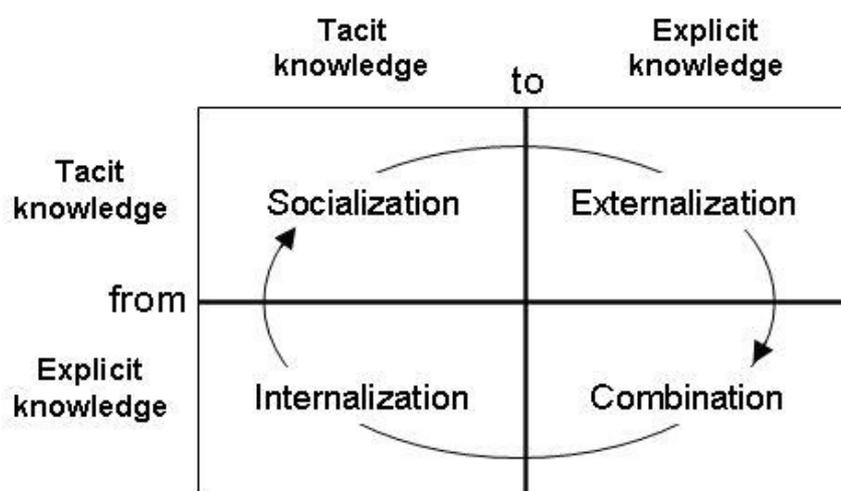


Figure 2.2 – Knowledge creation conversion modes (NONAKA, 1994).

Historically, organization's knowledge was undocumented, being represented through the skills, experience and knowledge of its professionals (i.e., tacit knowledge), which made its use and access limited and difficult (O'LEARY, 1998; RUS; LINDVALL, 2002). Knowledge Management (KM) aims to transform tacit and individual knowledge into explicit and shared knowledge. By raising individual knowledge to the organizational level, KM promotes knowledge propagation and learning, making knowledge accessible and reusable across the entire organization (O'LEARY, 1998; RUS; LINDVALL, 2002; SCHNEIDER, 2009). Therefore, when an organization implements KM, its experiences and knowledge are recorded, evaluated, preserved, designed and systematically propagated to solve problems (SCHNEIDER, 2009).

The literature presents different approaches, also known as KM cycles or models, that propose a set of activities in order to effectively promote KM initiatives (DALKIR, 2011). A KM cycle has activities that encompass, for example, capturing, creating, encoding, sharing, accessing, applying and reusing the individual, group and organizational knowledge within and between organizations. Dalkir (2011) summarized the major steps of a KM cycle in an integrated KM cycle (outlined in Figure 2.3), which consists of three main activities:

1. **Knowledge capture and/or creation:** Knowledge capture refers to the identification and subsequent codification of existing knowledge and know-how. Knowledge creation, in turn, is the development of new knowledge and know-how. In the transition from knowledge capture/creation to knowledge sharing and dissemination, knowledge content is assessed to verify if it provides sufficient value to the organization that it should be added to the store of intellectual capital.
2. **Knowledge sharing and dissemination:** Once it has been decided that the knowledge is of sufficient value, its content is contextualized in order to be understood and used. This involves maintaining a link between the knowledge and those knowledgeable about that content: the author or originator of the idea, subject matter experts, and also those who have garnered significant experience in making use of the content.
3. **Knowledge acquisition and application:** The knowledge management cycle is reiterated as users understand and decide to make use of content. The users will validate usefulness, help validate the scope of the content and, quite often, come up with new content, which can contribute to the next cycle iteration.

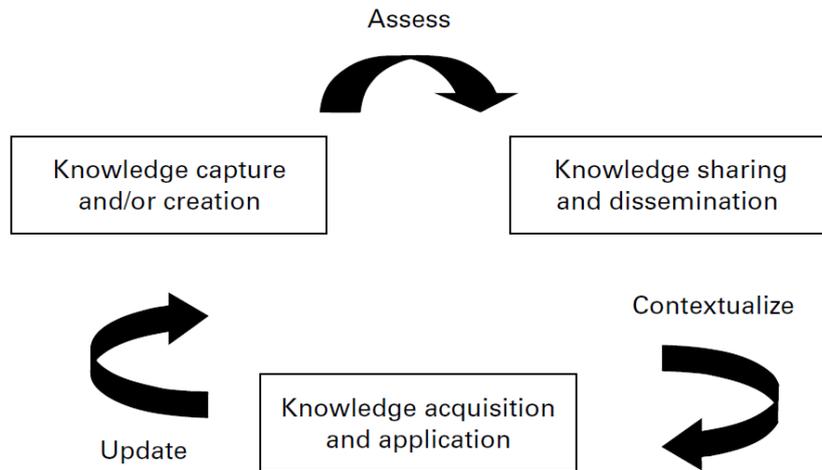


Figure 2.3 – Integrated KM cycle (DALIKIR, 2011).

Knowledge management depends on two aspects: **KM foundations**, which are broad organizational aspects that support KM, consisting of *KM infrastructure*, *KM mechanisms*, and *KM technologies*; and **KM solutions**, which refer to the ways in which specific aspects of KM (discovery, capture, sharing, and application of knowledge) can be accomplished, consisting of KM processes and KM systems (BECERRA-FERNANDEZ; SABHERWAL, 2010). *KM infrastructure* reflects the long-term foundation for knowledge management in an organization, including five major components: organization culture, organization structure, information technology infrastructure, common knowledge, and physical environment. *KM mechanisms*, in turn, are organizational or structural means used to promote knowledge management, which may (or may not) involve the use of information technology but involve some kind of organizational arrangement or social or structural means of facilitating KM. *KM technologies* are information technologies that can be used to facilitate knowledge management, i.e., they are intrinsically no different from information technologies, but they focus on knowledge management rather than information processing. *KM systems* are the integration of *KM technologies* and *KM mechanisms* that are developed to support *KM processes*, which are broad processes that help in discovering, capturing, sharing, and applying knowledge. The relation between these KM aspects is outlined in Figure 2.4.

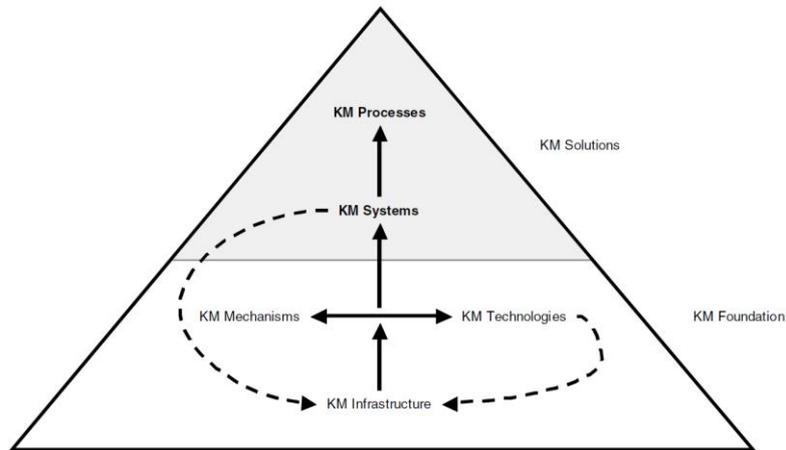


Figure 2.4 – An overview of KM Solutions and Foundation (BECERRA-FERNANDEZ; SABHERWAL, 2010).

In summary, KM works for explicitly and systematically managing knowledge, addressing knowledge acquisition, storage, organization, evolution, retrieval and usage. Being software development a knowledge-intensive process, KM has been applied in this context to support document management, competence management, experts identification, software reuse, learning and product and project memory, among others (RUS; LINDVALL, 2002).

HCI design can also be understood as a knowledge-intensive process, requiring effective mechanisms to collaboratively create and support a shared understanding about users, the system, its purposes, context of use and the design necessary for the user to achieve her goals. Thus, HCI design could take advantages of KM solutions. Furthermore, as HCI design and SE processes are intrinsically related, the integration of knowledge processes and methods may result in semantic interoperability conflicts (SEFFAH; GULLIKSEN; DESMARAIS, 2005). Ontologies can be helpful in this sense, providing a well-founded and consensual conceptualization of HCI design domain that can be used to enhance KM solutions.

2.3 Ontologies

An ontology is a formal and explicit specification of a shared conceptualization (STUDER; BENJAMINS; FENSEL, 1998). The conceptualization is an abstract and simplified view of the world which is intended to be represented for some reason. Every knowledge base, knowledge-based system or knowledge level agent is committed, either explicitly or implicitly, with one conceptualization (STAAB; STUDER, 2004).

Ontologies can be classified according to their generality level, as suggested by Guarino (1998). This classification defines four types of ontologies, as it is shown in Figure 2.5:

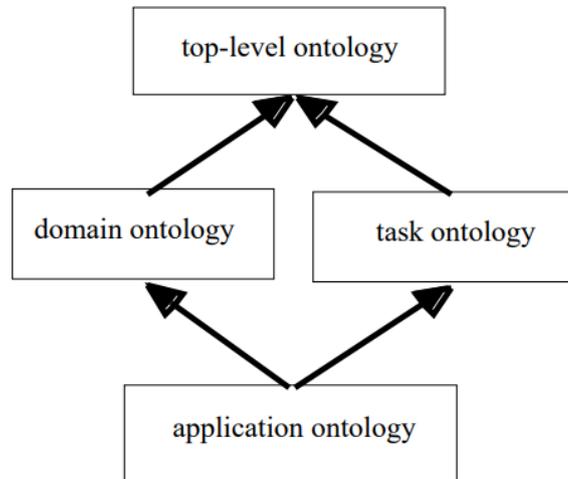


Figure 2.5 – Kinds of ontologies, accordingly to their generality level (GUARINO, 1998).

Top-level (or *foundational*) *ontologies* describe very general and domain independent concepts, such as space, time, matter, objects, events, actions etc. *Domain ontologies* describe the vocabulary related to a generic domain, such as health or automobiles, specializing terms from the upper-level ontology. *Task ontologies*, in turn, describe the vocabulary related to a generic task or activity, such as measurement, also specializing concepts from the upper-level ontology. Typically, this kind of ontology describes a process rather than a specific task. Finally, *application ontologies* describe concepts depending both on a particular domain and task, which commonly are specializations of both related ontologies.

Scherp *et al.* (2011) also classify ontologies according to their generality level. Like Guarino (1998), they consider foundational and domain ontologies. However, they define another type of ontology between them and organize ontologies in a three-layered architecture. *Core ontologies* are positioned between foundational and domain ontologies, providing a refinement to foundational ontologies by adding detailed concepts and relations in a specific area (such as service, process, organizational structure). Domain ontologies, in turn, can make use of or be based on foundational and core ontologies, by specializing their concepts. Falbo *et al.* (2013) additionally discuss that core ontologies lie in a continuous spectrum between foundational ontologies and domain ontologies, i.e., there are some core ontologies that are more general than others.

Another important distinction differentiates ontologies as conceptual models, called *reference ontologies*, from ontologies as computational artifacts, called *operational ontologies*

(GUIZZARDI, 2007). A reference ontology is constructed with the goal of making the best possible description of the domain in reality, representing a model of consensus within a community, regardless of its computational properties. Operational ontologies, in turn, are designed with the focus on guaranteeing desirable computational properties and, thus, are machine-readable ontologies.

For a complex domain, representing its knowledge as a single ontology results in a large and monolithic ontology that is hard to manipulate, use, and maintain (SUÁREZ-FIGUEROA *et al.*, 2012). On the other hand, representing each sub-domain in isolation is a costly task that leads to a very fragmented solution that is again hard to handle (RUY *et al.*, 2016). In such cases, building an ontology network (ON) is an adequate solution (SUÁREZ-FIGUEROA *et al.*, 2012). In an ON, ontologies are connected to each other through relationships. Two relationships can be highlighted: dependency and alignment. The former occurs when, in order to define its own model, an ontology refers to concepts and relations defined in another ontology (i.e., an ontology reuses concepts from another). The latter is a way to put different models in correspondence by establishing equivalency mappings between entities from different ontologies (i.e., the same as, a generalization of, a specialization of) (SUÁREZ-FIGUEROA *et al.*, 2012).

The ontologies proposed in this work were developed grounded in UFO (Unified Foundational Ontology) (GUIZZARDI, 2005) as foundational ontology and integrated to SEON (Software Engineering Ontology Network) (RUY *et al.*, 2016) and HCI-ON (Human-Computer Interaction Ontology Network) (COSTA *et al.*, 2020). The following subsections present fragments of UFO, SEON and HCI-ON that are relevant for this work.

2.3.1 UFO (Unified Foundational Ontology)

UFO is a foundational ontology that addresses many essential aspects for the conceptual modeling of the HCI design domain, such as agents, objects and mental aspects. UFO is composed of three main parts: UFO-A, an ontology of endurants (GUIZZARDI, 2005); UFO-B, an ontology of perdurants/events (GUIZZARDI *et al.*, 2013); and UFO-C, an ontology of social entities (both endurants and perdurants) built on top of UFO-A and UFO-B (GUIZZARDI; FALBO; GUIZZARDI, 2008). Figure 2.6 presents the fragment of UFO relevant to this work. Concepts that are directly used in this work are highlighted in purple in the figure. The concepts descriptions are based on (GUIZZARDI, 2005) and (GUIZZARDI; FALBO; GUIZZARDI, 2008). In the model description, UFO concepts are written in *italics*.

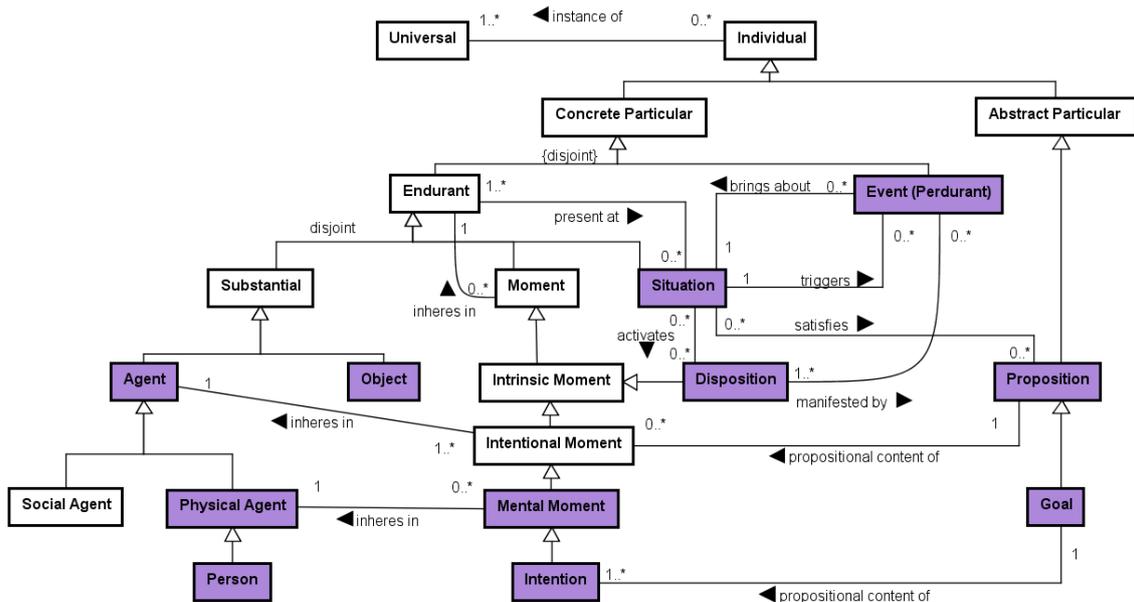


Figure 2.6 – UFO fragment relevant to this work.

The first fundamental distinction in UFO is between *Individuals* (particulars) and *Universals* (types). *Individuals* are entities that exist in reality, possess a unique identity (e.g., a person, a car) and instantiate *Universals*, which are patterns of features that can be realized in a number of different individuals (e.g., Person, Car). In this work, our focus is on *Individuals*. *Individuals* can be *Abstract* or *Concrete*. *Abstract Individuals* include numbers, sets, propositions, quality structures and quales, among others. *Concrete Individuals* are partitioned into *Endurants* and *Perdurants (Events)*. *Endurants* are said to be wholly present whenever they are present (e.g., a person), while *Events (Perdurants)* are individuals composed of temporal parts (e.g., a soccer match). Among the types of *Endurants*, two are detached: *Substantials* and *Moments*. *Substantials* are existentially independent individuals (e.g., an apple), while *Moments*, in contrast, denote properties of individuals. *Situations* are special types of *Endurants*, i.e., complex entities constituted by possibly many *Endurants* (including other *Situations*), that represent a portion of reality that can be comprehended as a whole, also known as a state of affairs (e.g., John has the flu and a fever). A *Situation* may trigger an *Event*, which brings about a new *Situation*. *Intrinsic Moments* are moments dependent on one single individual (e.g., an apple's color) and *Dispositions* are types of *Intrinsic Moments* that are only manifested in particular *Situations* on the occurrence of certain triggering *Events* (e.g., a magnet attracting property triggered after approaching to a metal object).

A basic distinction of *Substantials* is related to *Agents* and (non-agentive) *Objects*. *Agents* are agentive substantial individuals that can be *Physical Agents* (e.g., a person) or *Social Agents* (e.g., an organization, a society). *Objects* are non-agentive substantial individuals that can also

be physical (e.g., a book, a table) or social (e.g., money, language). *Agents* can bear special types of *Intrinsic Moments* named *Intentional Moments*. In this case, intentionality refers to the capacity of some properties of certain individuals to refer to possible situations in reality. Thus, *Intentional Moments* have a propositional content (*Proposition*), which is an abstract representation of a class of situations referred by that *Intentional Moment*. A proposition can be satisfied by a *Situation* when the *Situation* actually occurs in the real world. *Mental Moments* are specialization of *Intentional Moments* referring to mental components of a *Physical Agent*. A specific type of *Mental Moment* is an *Intention*, which is the proper representation of “intending something” that has a *Goal* as its propositional content.

2.3.2 SEON (*Software Engineering Ontology Network*)

SEON (RUY *et al.*, 2016) is an ontology network that describes various subdomains of the Software Engineering domain (e.g., software requirements, coding, testing, software measurement, etc.). It organizes its networked ontologies according to the layers defined by Scherp *et al.* (2011).

This work reuses concepts from three SEON ontologies, namely: Software Process Ontology (SPO) (BRINGUENTE; FALBO; GUIZZARDI, 2011), a core ontology that provides a conceptualization of software process, addressing aspects related to processes, activities, resources, stakeholders, artifacts and procedures; System and Software Ontology (SysSwO) (BRINGUENTE; FALBO; GUIZZARDI, 2011; DUARTE *et al.*, 2018), a core ontology about the nature of system and software, including, software artifacts, software constitution, software execution, computer system and hardware equipment; and Software Requirements Ontology (RSRO) (DUARTE *et al.*, 2018), a domain ontology that deals with concepts related to software requirements. Figure 2.7 shows the integrated view of the concepts from these ontologies that are relevant for this work. For simplification reasons, the model presents only the concepts directly reused in this work. In the figure, SPO concepts are presented in gray, SysSwO in green, RSRO in red and UFO concepts in white. Blue relationships represent the grounding of concepts in UFO. In the model description, UFO concepts are written in *italics* while **bold** is used in SEON concepts.

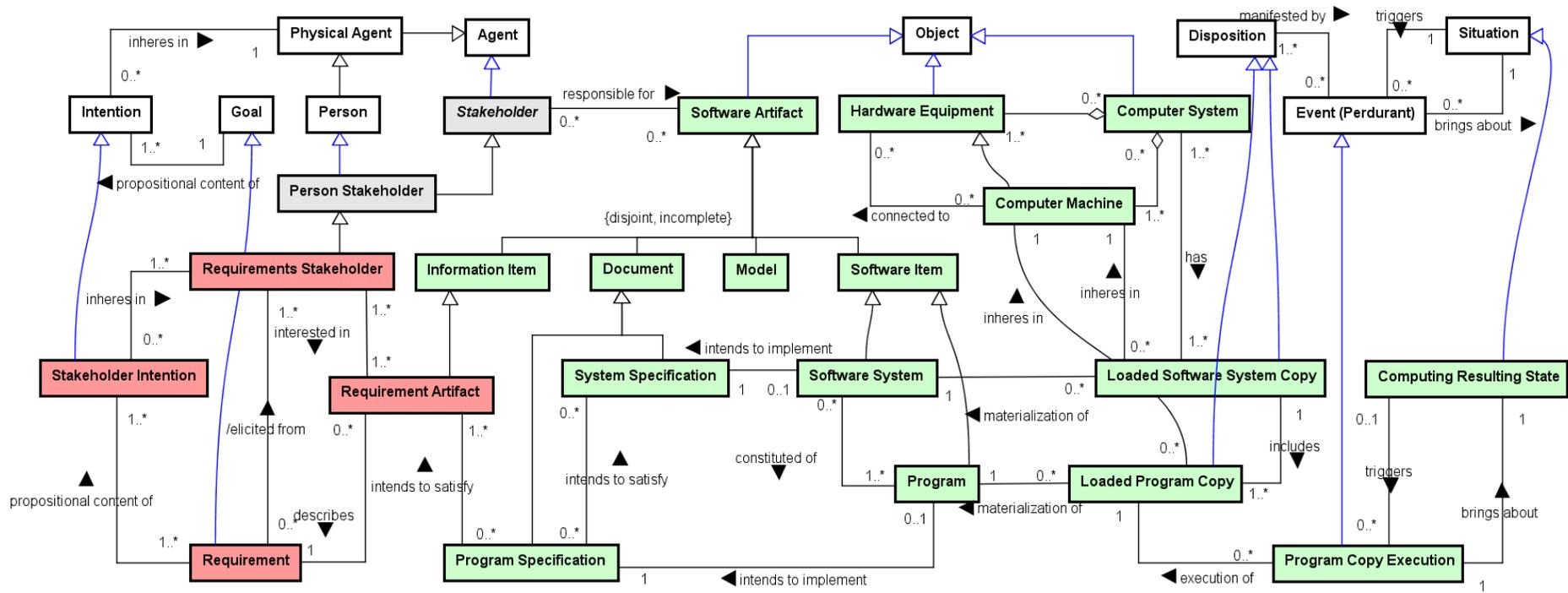


Figure 2.7 – SEON fragment relevant to this work.

A **Stakeholder** is an *Agent* interested in a particular software project. It can be a person, an organization or a team. In the first case, it is called **Person Stakeholder**. A Stakeholder may be responsible for **Software Artifacts**, which are *Objects* intentionally produced to serve a given purpose in the context of a software project or organization. **Software Artifacts** can be classified according to their nature. A **Software Item** is a piece of software, produced during the software process, not considered a complete product, but an intermediary result (e.g., a component). A **Document**, in turn, is any written or pictorial, uniquely identified information related to the software development, usually presented in a predefined format (e.g., a requirements document). An **Information Item** is a piece of relevant information for human use, produced or used by an activity (e.g., a component description, a bug report). A **Model** is a representation (abstraction) of a process or system from a particular perspective (e.g., a use case model, a class model).

A **Software System** (e.g., a system to buy airline tickets) (a subtype of **Software Item**) is constituted of **Programs** and intends to implement a **System Specification** (a subtype of **Document**). A **Program**, in turn, is also defined as a **Software Item**, a piece of software, produced during the software process, not considered a complete **Software System** (e.g., the system component to select available flights in a certain date). A program aims at producing a certain result through execution on a computer, in a particular way, given by the **Program Specification**, which is a **Document** describing structural and functional information about the **Program** with enough detail that would allow implementation and maintenance.

A **Hardware Equipment** is a *Physical Object* used to process, transform, store, display or transmit information or data. A **Hardware Equipment** that can run programs, process, transform and store data and information is a **Computer Machine**. A **Computer System** is a system containing one or more **Computer Machines** and other **Hardware Equipment** connected to them. A **Loaded Software System** is the materialization of a **Software System** (e.g., the system to buy airline tickets loaded in Mary's computer machine) as a complex *Disposition* inhering in a **Computer System**, including one or more **Loaded Program Copies**. A **Loaded Program Copy**, in turn, is the materialization of a **Program** (e.g., the component to select available flights in a certain date loaded in Mary's computer machine) as a *Disposition* manifested by a **Program Copy Execution** (*Event*). A **Program Copy Execution** (e.g., the execution of the program copy to show flights available in a certain date) brings about a **Computer Resulting State** (e.g., a set of flights), a *Situation* involving properties of the **Computer Machine** in which the **Loaded Program Copy** inheres, as well

as of entities residing in that **Computer Machine** (including the **Loaded Program Copy** itself). A **Computer Resulting State** may also trigger another **Program Copy Execution**.

A **Requirement** is a *Goal* in the sense of UFO, i.e., the propositional content of an *Intention (Mental Moment)* that inheres in a **Requirements Stakeholder**. When a **Requirement** is recorded in some kind of document, there is a **Requirement Artifact** describing that **Requirement**. A **Requirement Artifact** is an **Information Item** that is responsible for keeping relevant information for human use. **Requirements** are connected to implemented software through the following relation: a **Program Specification** intends to satisfy some **Requirement Artifacts**. Thus, a **Program** that intends to implement a **Program Specification** also intends to satisfy these **Requirement Artifacts**.

2.3.3 HCI-ON (*Human-Computer Interaction Ontology Network*)

HCI-ON (COSTA *et al.*, 2020) is an ontology network that has been developed and addresses several Human-Computer Interaction aspects, such as context of use, evaluation, UI types & elements, among others. Since several HCI concepts are related to SE, HCI-ON is integrated to SEON (RUY *et al.*, 2016) and is also organized in three layers (SCHERP *et al.*, 2011).

Figure 2.8 presents an overview of HCI-ON architecture and its connection with SEON, including the ontologies relevant to this work. Both HCI-ON and SEON adopt UFO in the foundational layer, keeping the same foundation on both ONs concepts and making it easier to connect them. In the figure, each circle (network's node) represents HCI-ON and SEON core or domain ontology. Dotted circles represent HCI-ON ontologies under development. Red arrows (directed arcs) represent dependency relationships from HCI-ON to SEON. HCI-ON dependency to SEON core ontologies are denoted by red solid arrows, while to SEON domain ontologies are denoted by red dotted arrows.

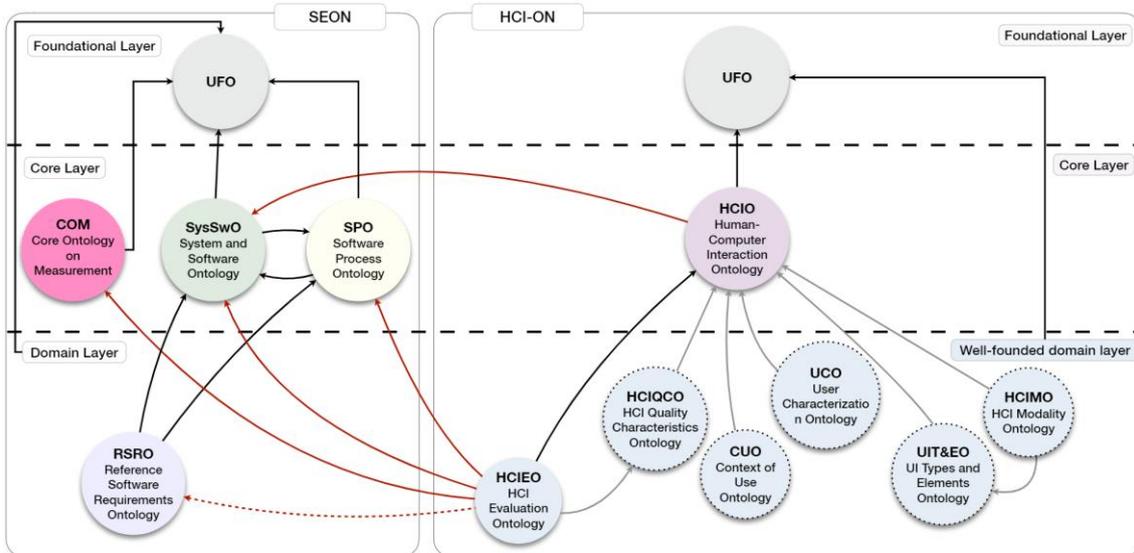


Figure 2.8 – Overview of HCI-ON and its connections with SEON (adapted from (COSTA, 2021))

This work reuses concepts from the Human-Computer Interaction Ontology (HCIO) (COSTA, 2021), a core ontology of HCI-ON. HCIO aims to clarify the main HCI notions and establish an explicit common and shared conceptualization about the HCI phenomenon. Being at the heart of HCI-ON, HCIO is organized in three sub-ontologies: (i) *Interactive Computer System* sub-ontology focuses on what an interactive computer system is and its constituent elements, including the user interface; (ii) *User* sub-ontology focuses on the user and intentional or unintentional actions performed by users when interacting with an interactive computer system; (iii) *Human-Computer Interaction* sub-ontology links concepts from the other two sub-ontologies to define what a human-computer interaction is. Figure 2.9 presents a fragment of HCI-ON containing concepts that are relevant to this work. In the figure, *Interactive Computer System* sub-ontology concepts are presented in yellow, *User* sub-ontology in magenta and *Human-Computer Interaction* sub-ontology in light blue (the colors for SysSwO and UFO are kept the same as from Figure 2.7). In the model description, UFO concepts are written in *italics*, **bold** is used in SEON concepts and HCI-ON concepts are underlined.

Interactive Computer System is a subtype of **Computer System**, and like that, it combines hardware and software. Concerning hardware, the striking feature of an Interactive Computer System is that it has a User Interface, a complex *Object* composed of Input Equipment and Output Equipment, which are devices (**Hardware Equipment**) connected to the **Computer Machine**. Regarding software, an Interactive Computer System has a set of Interactive Software Systems materialized as Loaded Interactive Software System Copies inhering in its **Computer Machine**. An Interactive Software System is a **Software System** constituted of **Programs**, of which some of them deal with aspects related to the User Interface and, thus, are instances of User Interface Program. Hence, an Interactive Computer System has a User Interface and copies of User Interface Programs loaded in its **Computer Machine** (Loaded User Interface Program Copies), handling its User Interface.

User is a role played by a *Person* that participates in a human-computer interaction. Such participation is said a User Participation, which can be either intentional or unintentional. Intentional participations are caused by *Intentions* (User Intentions) that inhere in the User. As an *Intention*, User Intention has a *Goal* (more specifically, a User Goal) as its propositional content. In another classification, which considers the nature of participations and is orthogonal to the one discussed above (i.e., they can also be either intentional or unintentional), User Participations are classified into two disjoint types: User Initiated Participation and User Interpretation. User Initiated Participation refers to an act performed by the user making an input in the system (e.g., to click a button). User Interpretation, in turn, regards interpreting a state of the system (e.g., to interpret what happened after the button was clicked).

A User Initiated Participation is performed using one or more Input Equipment and, as a result, a User Input Resulting State is achieved (e.g., the button is clicked). User Input Resulting State is a *Situation* representing the data entered by the user before any program execution. This situation triggers **Program Copy Executions** that bring about a **Computing Resulting State** (internal computer state), which, in turn, can trigger other **Program Copy Executions**. Some **Program Copy Executions** can bring about a special type of **Computing Resulting State**, the one that is perceivable by the user, said User Observable State (e.g., a message is shown because the button was clicked). A User Observable State, thus, triggers User Interpretation, which may evaluate the achievement of User Goals.

2.4 Ontologies in HCI Design context

The literature discusses several cases for employing ontologies in HCI Design context. Paulheim and Probst (2010) presented a survey of state-of-the-art approaches in which the development process or the usability of a user interface has been improved by employing ontologies. Based on the survey results, they elicited three main purposes for using ontologies to enhance user interfaces: (i) improving visualization capabilities, (ii) improving interaction possibilities, and (iii) improving the development process of the user interface.

(i) and (ii) usually aim to support users of interactive systems, employing operational ontologies at run-time, sometimes combined with other tools (e.g., reasoners), to change interface or interaction aspects of the system (PAULHEIM; PROBST, 2010). For example, they can be used to build adaptive user interfaces, where the interface components may change based on specific user needs (e.g., users with color blindness) (KULTSOVA *et al.*, 2017). Another case in this context is to support self-explanatory user interfaces, in which ontology-based formalizations are used to create help texts and visual hints at run-time (KOHLHASE; KOHLHASE, 2009). On the other hand, approaches that aim (iii) often occur at design time and the end user of the system does not see the ontologies, nor interact with them (PAULHEIM; PROBST, 2010). A classic use in this context is to support the creation of metamodels in a model driven approach (SUÀREZ; JÚNIOR; DE BARROS, 2004). Ontologies have also been used to support annotation and classification in repositories of user interface components and repositories of usability patterns, enhancing the search for elements that fit designers' needs in different contexts (HAPPEL *et al.*, 2006; HENNINGER; KESHK; KINWORTHY, 2004).

With regard to the scope addressed by ontologies used in HCI design context, the concepts are usually focused either on the interaction between user and system or on the user interface and its components. Hence, they are not focused on describing the design of the human-computer interaction itself. One example focused on the interaction is the formal ontology proposed by Silva *et al.* (2017), which describes interactive behaviors on user interfaces, aiming to support test automation of interactive systems functional requirements. The formal ontology *UI2Ont* (PAULHEIM; PROBST, 2013), in turn, focuses on the user interface, addressing general concepts that exist in the user interface domain (such as components and activities) and detailed taxonomies of those concepts, i.e., a categorization of component types etc. On the other hand, some ontologies focus on HCI design applied to specific contexts, such as web design (BAKAEV; GAEDKE, 2016; BAKAEV; AVDEENKO, 2010), design for haptic devices (MYRGIOTI; BASSILIADES; MILIOU,

2013) and design for gesture based interactions (CHERA; TSAI; VATAVU, 2012). Therefore, none of the ontologies found in the literature provide a comprehensive conceptualization about HCI design. Moreover, they are not concerned with representing mental aspects of HCI design, which are very important to make explicit the connection between the choices made when designing the HCI of an interactive system and the resulting HCI design.

2.5 Concluding Remarks

This chapter addressed the main background related to this work. First, we discussed some HCI design aspects, explaining how the human-computer interaction occurs and what HCI design is. HCI design focuses on how to design the interactive computer system to support users to achieve their goals through the interaction between them and the system (SUTCLIFFE, 2014). Then, we presented the main notions of KM.

After that, we discussed basic concepts about ontologies and presented fragments of the foundational ontology and the ontology networks used in this work, namely UFO, SEON and HCI-ON. We also discussed how ontologies have been used in the HCI design context and pointed out the lack of ontologies that provide a comprehensive conceptualization about HCI design, covering not only related artifacts, but also mental aspects.

HCI design is characterized as a knowledge-intensive process involving knowledge transfer challenges and semantic interoperability issues that can be addressed by the combination of ontologies with knowledge management solutions.

Aiming to investigate the use of KM in HCI design, we performed a systematic mapping and a survey, which are addressed in the next two chapters.

Chapter 3

Systematic Mapping: KM in HCI Design

This chapter addresses a mapping study that investigated the use of KM in HCI design in the literature and presents its main results. It is organized as follows: Section 3.1 presents the chapter introduction; Section 3.2 addresses the research protocol; Section 3.3 summarizes the obtained results; Section 3.4 discusses the results; Section 3.5 presents some of the limitations of the study; and Section 3.6 presents the chapter concluding remarks.

3.1 Introduction

Considering the challenges of designing interactive systems, mainly due to the diversity of knowledge and people involved, and the potential of KM to help address those challenges, we decided to investigate the use of KM in HCI design. First, we searched for secondary studies addressing the research topic. Since we did not find any, we decided to perform a systematic mapping in the literature.

A mapping study is a secondary study designed to give an overview of a research area through classification and counting contributions in relation to the categories of that classification. It makes a broad study in a topic of a specific theme and aims to identify available evidence about that topic (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). Moreover, the panorama provided by a mapping study allows identifying issues in the researched topic that could be addressed in future research. We followed the process defined in (KITCHENHAM; CHARTERS, 2007), which comprises three phases:

1. *Planning*: In this phase, the topic of interest, study context and object of the analysis are established. The research protocol to be used to perform the research is defined, containing all the necessary information for a researcher to perform the research: research questions, sources to be searched, publication selection criteria, procedures for data storage and analysis and so on. The protocol must be evaluated by experts and tested to verify its feasibility, i.e., if the results obtained are satisfactory and if the protocol execution is viable in terms of time and effort. Once the protocol is approved, it can be used to conduct the research.
2. *Conducting*: In this phase, the research is performed according to the protocol. Publications are selected and data are extracted, stored and quantitatively and qualitatively analyzed.

3. *Reporting*: In this phase, the produced research results are recorded and made available to potential interested parties.

3.2 Research protocol

This section presents the protocol used in the mapping study. It was defined gradually, being tested with an initial set of publications and then refined until we reached the final protocol, which was evaluated by another researcher, resulting in the protocol used in the study and presented in this section.

The study *goal* was to investigate the use of KM in HCI design context. For achieving this goal, we defined the research questions presented in Table 3.1.

Table 3.1 – Systematic Mapping: research questions and their rationale.

ID	Research Question	Rationale
RQ1	<i>When and where have publications been published?</i>	Give an understanding on when and where (journal / conference / workshop) publications about KM in HCI design context have been published.
RQ2	<i>Which types of research have been done?</i>	Investigate which type of research is reported in each selected publication. We consider the classification defined in (WIERINGA <i>et al.</i> , 2005). This question is useful to evaluate the maturity stage of the research topic.
RQ3	<i>Why has KM been used in the HCI design context?</i>	Understand the purposes and reasons of using KM in the HCI design and verify if there have been predominant motivations.
RQ4	<i>Which knowledge has been managed in the HCI design context?</i>	Investigate which knowledge items have been managed in the HCI design context, aiming to verify if some of them have been managed more frequently and if there has been more interest in certain HCI aspects.
RQ5	<i>How is the managed knowledge related to the HCI design process?</i>	Understand, in the context of the HCI design process, from where the managed knowledge has been coming from and where it has been used.
RQ6	<i>How has KM been implemented in the HCI design context?</i>	Investigate how KM has been implemented in HCI context in terms of the adopted technologies.
RQ7	<i>Which benefits and difficulties have been noticed when using KM in the HCI design context?</i>	Identify benefits and difficulties of using KM in HCI design context and analyze if there is relation between them.

RQ1 and RQ2 are common systematic mapping questions that provide a general panorama of the research topic. The other questions aim to investigate why (RQ3 and RQ7), how (RQ4 and RQ6) and when (RQ5) KM has been used in HCI design, which are important questions to provide an understanding of the research topic.

The *search string* adopted in the study contains two groups of terms joined with the operator AND. The first group includes terms related to *HCI design*. The general term “Human-Computer Interaction” was used to provide wider search results. The second group includes terms related to *Knowledge Management*. Within the groups, we used the OR operator

to allow synonyms. The following search string was used: ("*human-computer interaction*" OR "*user interface design*" OR "*user interaction design*" OR "*user centered design*" OR "*human-centered design*" OR "*UI design*" OR "*HCI design*") AND ("*knowledge management*" OR "*knowledge reuse*" OR "*knowledge sharing*"). For establishing the string, we performed tests using different terms, logical connectors, and combinations among them, and selected the string that provided better results in terms of the number of publications and their relevance (i.e., the number of publications returned by the search string and, considering a sample, the inclusion of the really relevant ones for the study). If a new term added to the search string resulted in a much larger number of returned publications, without adding new relevant ones to the study, then that term was not considered in the search string. In that sense, more restrictive strings excluded important publications identified during the informal literature review that preceded the study. More comprehensive strings (e.g, those including “usability”) returned too many publications out of the scope of interest.

The search was performed in four **sources**, namely Scopus, Science Direct, Engineering Village and Web of Science. We selected these sources because Scopus is one of the largest databases of peer-reviewed literature. It indexes papers from other important sources such as IEEE and ACM, providing useful tools to search, analyze and manage scientific research. Complementarily, to increase coverage, we selected Science Direct, Engineering Village and Web of Science, which are also widely used in secondary studies recorded in the literature and on other experiences in our research group.

Publications selection was performed in five steps. In *Preliminary Selection and Cataloging (S1)*, the search string was applied in the search mechanism of each digital library used as source of publications (we limited the search scope to title, abstract and keywords metadata fields). After that, in *Duplications Removal (S2)*, publications indexed in more than one digital library were identified and duplications were removed. In *Selection of Relevant Publications - 1st filter (S3)*, the abstracts of the selected publications were analyzed considering the following inclusion (IC) and exclusion (EC) criteria: (IC1) the publication addresses KM in the HCI design context; (EC1) the publication does not have an abstract; (EC2) the paper was published only as an abstract; (EC3) the publication is not written in English; (EC4) the publication is a secondary study, a tertiary study, a summary, an editorial or a tutorial. In *Selection of Relevant Publications - 2nd filter (S4)*, the full text of the publications selected in S3 were read and analyzed considering the cited inclusion and exclusion criteria. In this step, to avoid study repetition, we considered another exclusion criterion: (EC5) the publication is an older version of an already selected publication. When the full text of a publication was

not available either from the Brazilian Portal of Journals, from other Internet sources or by contacting its authors, the publication was also excluded (EC6). Publications that met one of the six cited exclusion criteria or that did not meet the inclusion criteria IC1 were excluded. Finally, in *Snowballing* (S5), as suggested in (KITCHENHAM; CHARTERS, 2007), the references of publications selected in S4 were analyzed by applying the first and second filters and, the ones presenting results related to the research topic were included in the study.

We used the StArt tool¹ to support publications selection. To consolidate data, publications returned in the publication selection steps were cataloged and stored in spreadsheets. We defined an id for each publication and recorded the publication title, authors, year, and vehicle of publication. Data from publications returned in S4 and S5 were extracted and organized into a data extraction table oriented to the research questions. The spreadsheets produced during the study can be found in <http://bit.ly/Mapping-KM-in-HCI-design>.

The mapping was conducted by four researchers. The first and second researchers performed publication selection and data extraction. The third and fourth researchers (supervisors of this dissertations) reviewed both. Once data has been validated, the first and the second researchers carried out data interpretation and analysis, and again third and fourth researchers reviewed the results. Discordances were discussed and resolved. Quantitative data were tabulated and used in graphs and statistical analysis. Finally, the four researchers performed qualitative analysis considering the findings, their relation to the research questions and the study purpose.

3.3 Results

The study considered papers published until October 2020. Searches were conducted for the last time in November 2020. Figure 3.1 illustrates the followed process and the number of publications selected in each step.

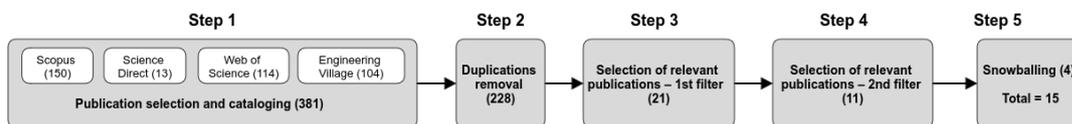


Figure 3.1 – Publications selection process

In the 1st step, as a result of searching the selected sources, a total of 381 publications was returned. In the 2nd step, we eliminated duplicates, achieving 228 publications

¹ <http://bit.ly/StArt-tool>

(reduction of approximately 40%). In the 3rd step, we applied the selection criteria over the abstract, resulting in 21 papers (reduction of approximately 91%). At this step, we only excluded publications that were clearly unrelated to the subject of interest. In case of doubt, the paper was taken to the next step. In the 4th step, the selection criteria were applied considering the full text, resulting in 11 publications (reduction of approximately 48%). Finally, in the 5th step, we performed snowballing technique by checking the references of the 11 selected publications and identified 4 more publications, which in total added up to 15 publications. When analyzing the publications to identify the KM approaches applied in HCI design context, we noticed that some publications addressed complementary works from a same research group. Hence, we considered complementary works as a single KM approach when extracting data about RQs 3, 4, 5, 6 and 7. Table 3.2 shows the list of identified KM approaches, their description and corresponding publications. Two papers were grouped into a KM approach and three other papers were grouped in another KM approach. Thus, we considered a total of 12 different KM approaches found in 15 publications. Along this and the next section we refer to the approaches by using the id listed in the table. After Table 3.2, we present the data synthesis for each research question. Further information about the selected publications, including detailed extracted data, can be found in <http://bit.ly/Mapping-KM-in-HCI-design>.

Table 3.2 – Selected publications.

ID	Approach	Brief description	Ref.
#01	<i>Trading off usability and security in user interface design through mental models</i>	Proposes the development of an Organizational Mental Model through knowledge transfer and transformation, using collaborative brain power from various knowledge constellations to design.	(MOHAMED; CHAKRABORTY; DEHLINGER, 2017)
#02	<i>Knowledge management challenges in collaborative design of a Virtual Call Centre</i>	Proposes a knowledge-based system with the following functionalities: (a) storing design primitives and formal knowledge in an online library; (b) preserving procedures and rules that proved successful in past design problems; (c) formal modeling of knowledge elements which might be applicable for usability improvements; (d) providing multiple mechanisms for knowledge acquisition, preserving, transfer and sharing.	(SIKORSKI <i>et al.</i> , 2011)
#03	<i>Applying knowledge management in UI design process</i>	Defines a process to automate the transformation of a task description into an interaction description. First, it identifies and uniformizes existing knowledge about UI design process using knowledge classification techniques. Then, captured knowledge is represented in the form of ontologies, deriving a Task Metamodel and an Interaction Metamodel. This extracted knowledge is integrated to design defining a transformation of task description into interaction description using an intermediate model between them and a two-step transformation.	(SUÀREZ; JÚNIOR; DE BARROS, 2004)

Table 3.2 (continuation) – Selected publications.

ID	Approach	Brief description	Ref.
#04	<i>A knowledge management tool for speech interfaces</i>	Proposes a knowledge-based system to help developers of speech driven interfaces learn with previous design solutions. These solutions are collected, made accessible and divided into categories regarding their content type. Solutions with corresponding structures are clustered and compared within their own category, providing to designers a suggestion mechanism based on their desired kind of solution. There is also a ranked suggestion mechanism of design elements based on available design material and design guidelines.	(BOUWMEESTER, 1999)
#05	<i>Design knowledge reuse based on visualization of relationships between claims</i>	Presents a tool that aims to improve design and knowledge acquisition by exploring relationships between claims. It allows a better search and retrieval mechanism to a design knowledge repository, which is obtained by applying KM strategies (generalize, classify, store, retrieve) to claims.	(WAHID, 2006; WAHID et al., 2004)
#06	<i>Design knowledge reuse and notification systems to support design in the development process</i>	Presents a system connected to a design knowledge repository based on claims. It allows teams to leverage knowledge from previous design efforts by searching for reusable claims relevant to their current project and to extend the repository by updating existing claims and creating new ones.	(CHEWAR et al., 2004; CHEWAR; MCCRICKARD, 2005; SMITH; BOHNER; MCCRICKARD, 2005)
#07	<i>Exploring knowledge processes in user-centered design process</i>	Proposes a conceptual framework that guides the design process based on five propositions: (1) designers and users should be actively included as actors in the process, since they both have knowledge needed to successful design; (2) this knowledge possessed by them is context-specific; (3) there is useful knowledge that has not been articulated by both users and designers and, therefore (4) knowledge processes transforming tacit knowledge into explicit knowledge by users and designers are linked and should be combined; and finally, (5) resulting knowledge obtained along the process is embedded into concepts, products or services.	(STILL, 2006)
#08	<i>Lessons learnt from an HCI repository</i>	Concerns the implementation of a knowledge repository using Windows Help Files. It is maintained by a group within the organization that receives content updates from the team and properly insert this new material into the repository. New versions are released from time to time and distributed as physical copies to be installed in each computer.	(WILSON; BORRAS, 1998)
#09	<i>A pattern language approach to usability knowledge management</i>	Presents a KM system that used principles of use case writing and pattern languages to describe problems found in user testing sessions and the following solutions to them. Patterns can be retrieved by forms with filters, text search and database queries. Filters include goals and subgoals, being useful respectively to show all problems related to a specific user goal and possible solutions and to provide insights of what interactions or devices have been problematic regardless of user goal.	(HUGHES, 2006)
#10	<i>An expert system for usability evaluations of business-to-consumer e-commerce sites</i>	Proposes a knowledge-based system to help on e-commerce usability evaluations. A knowledge engineer is responsible for acquiring and representing knowledge, eliciting knowledge from textual, non-live sources of expertise about design guidelines that affect usability of 11 e-commerce elements. The elicited knowledge is consolidated and presented in a form of rules in the expert system.	(GABRIEL, 2007)

Table 3.2 (continuation) – Selected publications.

ID	Approach	Brief description	Ref.
#11	<i>A framework for developing experience-based usability guidelines</i>	Presents a KM system to manage design guidelines contextualized by usability examples. The system allows designers to describe their current problem and requirements and then search for cases with similar characteristics. They can also follow hyperlinks to more general guidelines, which also point to other cases and search from a list of hierarchically arranged guidelines and follow other related guidelines and cases. The system is initially seeded with organization-wide usability guidelines and is updated as new projects are developed.	(HENNINGER; HAYNES; REITH, 1995)
#12	<i>Prototype evaluation and redesign: structuring the design space through contextual techniques</i>	Proposes a method based on contextual inquiry and brainstorming to identify usability issues in interface evaluations and derive proper design solutions to them. First, interface evaluation sessions are conducted with users, when they share their perceptions while interacting with a high-fidelity prototype of the system. Those sessions are recorded and, later, relevant comments are transcribed into usability flaws. In a second moment, there are brainstorm meetings where developers, designers and HCI specialists propose design solutions to the previously identified usability flaws.	(SMITH; DUNCKLEY, 2002)

Publication year and type (RQ1): Figure 3.2 shows the distribution of the 15 selected publications over the years and their distribution considering the publication type. Papers addressing KM in HCI design context have been published since 1995 in Journals and Conferences (no Workshop publications were found). Conferences have been the main forum, encompassing 73.3% of the publications (11 out of 15). Four papers (26.78%) were published in journals.

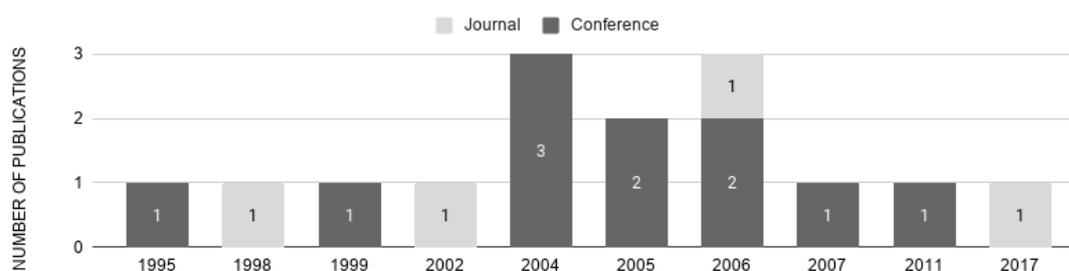


Figure 3.2 – Publications over the years.

Research Type (RQ2): Figure 3.3 presents the classification of the research types (according to the classification proposed in Wieringa *et al.* (2005)) reported in the 15 selected publications. 13 publications (86.7%) propose a solution to a problem and argue for its relevance. Thus, they were classified as *Proposal of Solution*. Five of them (33.3%) also present some kind of evaluation, being one (6.7%) evaluated in practice (i.e., also classified as *Evaluation Research*), and four (26.7%) investigating the characteristics of the proposed solution not yet implemented in practice (i.e., *Validation Research*). One publication (6.7%)

refers exclusively to *Evaluation Research*, discussing the evaluation of KM an industrial setting, and another is a *Personal Experience Paper*, reporting the experience of the authors in a particular project in the industry.

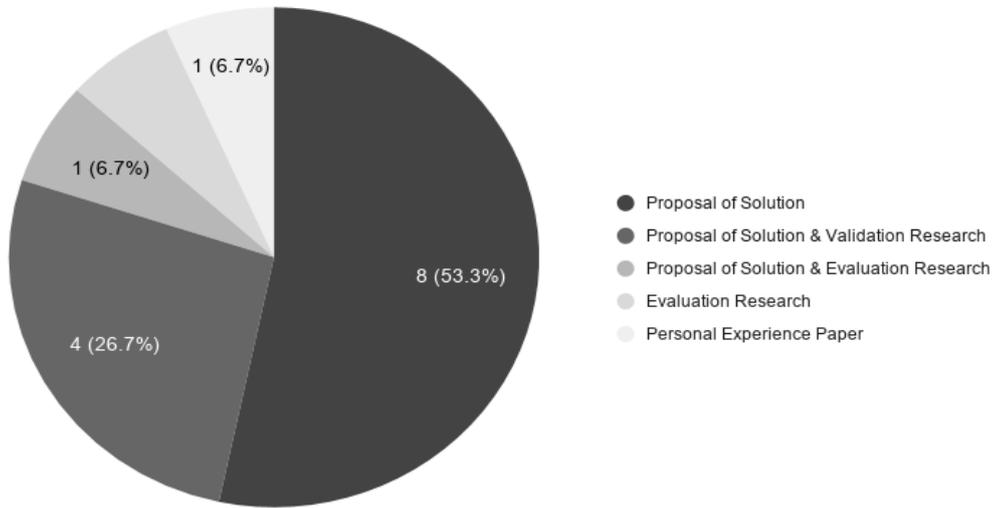


Figure 3.3 – Research type of the identified publications.

Motivation for using KM in HCI design (RQ3): we identified six reasons for using KM in HCI design, as shown in Table 3.3. Some approaches presented more than one motivation, thus the total sum is greater than 12. Nine approaches (75%) use KM to improve product quality, most of them concerning usability. These approaches aim to provide benefits related to the quality of the interactive system in terms of its interaction with users. For example, approach #11 is proposed to help developers to design effective, useful and usable applications. Approach #01, in turn, aims to improve alignment between design features and users’ requirements. Seven approaches (58.3%) are motivated by improving one or more aspects related to the HCI design process, namely: effort, time and cost. From these, reducing effort is highlighted. Five approaches (41.7%) use KM to reduce design effort, mainly by not depending on internal usability experts to perform HCI design activities. Approach #02, for example, applied KM to decrease the need for experts to support the design team with their knowledge and experience, due to lack of knowledge to be reused. Approaches #04, #05 and #08 were motivated by reducing HCI design time through the reuse of previous solutions implemented for similar problems. Reducing costs in the HCI design process was the motivation for approaches #05 and #10, which focus on minimizing the involvement of external usability experts in the process and conducting usability evaluation more effectively. Approach #06 aimed to improve design team performance by

providing support for team coordination and collaboration. This approach also aimed to improve HCI learning to the students involved in the project.

Table 3.3 – Motivations for using KM in HCI design.

Motivation	Approaches	Total
Improve product quality	#01, #02, #04, #05, #06, #07, #10, #11, #12	9
Reduce design effort	#02, #03, #08, #09, #10	5
Reduce design time	#04, #05, #08	3
Reduce design cost	#05, #10	2
Improve design team performance	#06	1
Improve HCI design learning	#06	1

Managed knowledge in HCI design (RQ4): Analyzing the publications, we identified 24 different types of knowledge items managed by the KM approaches, as shown in Table 3.4. Some items are shown in the same line to save space. The most common knowledge items have been *Design Guidelines* and *Design Solutions*, addressed by four approaches, followed by Test Results, addressed by three approaches. We noticed that, in the context of HCI design, KM approaches have dealt with only one (#10) or two (#01, #03, #05, #06, #09, #11 and #12) different knowledge items.

Table 3.4 – Managed knowledge items.

Knowledge Item	Approaches	Total
Design Guidelines	#04, #08, #10, #11	4
Design Solutions	#02, #04, #07, #08	4
Test Results	#02, #04, #12	3
Claims	#05, #06	2
Design Features	#01, #12	2
Design Patterns	#09, #11	2
Lessons Learned	#04, #08	2
Usability Measures	#02, #08	2
Claims Relationships	#05	1
Design Changes	#06	1
Design Feature Checklists; Design Methods; Design Processes; Design Standards; Design Templates; Interface Objects	#08	1
Interaction Model; Task Model	#03	1
Scenarios; Test Scenarios	#02	1
User Knowledge; User Needs	#07	1
User Requirements	#01	1
User Tasks	#09	1

We identified four different HCI aspects addressed by the identified KM approaches. The main aspect is *Usability*, which is treated in all the identified approaches. Two approaches (#03 and #08) also address *Ergonomics*. #03 and #04 focus on particular types of design or interface. The former focuses on *Task-based Design* while the latter on *Speech Driven Interfaces*.

Figure 3.4 shows the HCI aspects addressed in the identified KM approaches. The sum exceeds 12 because some approaches address more than one aspect.

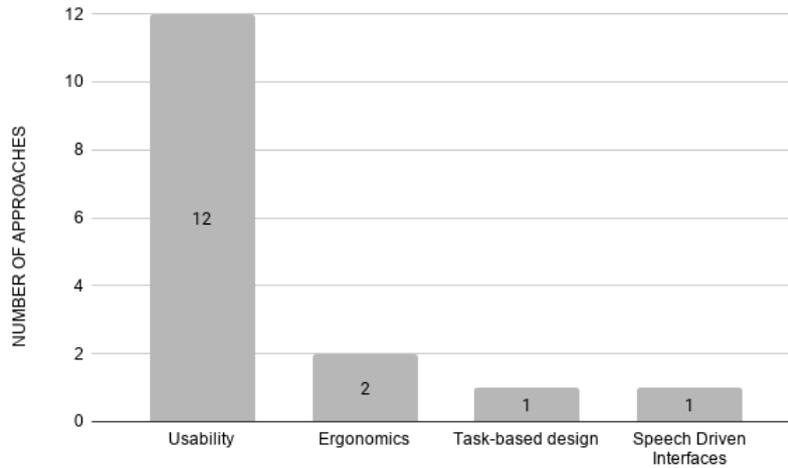


Figure 3.4 – HCI aspects addressed in KM approaches.

When knowledge is captured and used (RQ5): Table 3.5 shows when HCI design knowledge has been captured and when it has been used along the HCI design process. Three approaches capture and use knowledge along the whole process. Eight approaches (66.7%) use knowledge when producing design solutions. A smaller number (six, 50%) capture knowledge in this activity. The behavior is the opposite in design evaluation: there are more approaches capturing (five, 41.7%) than using (three, 25%) knowledge in this activity. Only one (8.3%) approach captures knowledge during requirements specification.

Table 3.5 – Capture and use of knowledge along the HCI design process.

Activity	Knowledge Capture	Knowledge Use
Specify requirements	1 (#01)	0
Produce design solutions	6 (#02, #03, #04, #07, #10, #11)	8 (#01, #02, #03, #04, #07, #09, #11, #12)
Design Evaluation	5 (#02, #04, #09, #10, #12)	3 (#02, #09, #10)
Whole cycle	3 (#05, #06, #08)	3 (#05, #06, #08)

Technologies used in KM approaches (RQ6): Table 3.6 shows the technologies (systems, methods, tools, theories, etc.) used in the analyzed KM approaches. The most common technologies were *knowledge-based systems* and *knowledge repositories*, which are used in three approaches. For example, #04 proposes a knowledge-based system to help developers of speech driven interfaces learn with previous design solutions. #08, in turn, proposes the implementation of a knowledge repository using Windows Help Files.

Knowledge management systems and *knowledge-based analysis* were used in two approaches. A knowledge management system is proposed in #09 to describe problems detected in user test sessions and the respective solutions and in #11 to describe design problems and requirements and then search for usability examples with similar characteristics and hyperlinks to more general related guidelines. Knowledge-based analysis, in turn, was used in #03 and #07 combined with other technologies, such as *ontology* and *model transformation* (#3) and *conceptual framework* (#7).

Other technologies such as *brainstorming*, *contextual inquiry*, *heuristic evaluation* and *mental models* were used in only one KM approach.

Table 3.6 – Technologies used in KM approaches in HCI design context.

Technology	Approaches	Total
Knowledge-based System	#02, #04, #10	3
Knowledge Repository	#05, #06, #08	3
Knowledge Management System	#09, #11	2
Knowledge-based Analysis	#03, #07	2
Ontology; Model Transformation	#03	1
Conceptual Framework	#07	1
Contextual Inquiry; Brainstorming-based Technique	#12	1
Mental Model; Internalization Awareness; Observation; Behavioral Interviews; Absorptive Capacity; Heuristic Evaluation	#01	1

Benefits and challenges of using KM in HCI design (RQ7): Table 3.7 and Table 3.8 summarize the benefits and difficulties reported in the publications. Two approaches (#04 and #10) did not report any benefit or challenge in using KM in HCI design. Considering the 10 other approaches, it can be noticed that, in general, more benefits than difficulties were reported.

The most reported benefit was to enable replicability of domain or context knowledge. For example, #07 reached a wide scope applicability because of the common conceptualization proposed as a conceptual framework. On the other hand, the most reported difficulty was that knowledge is often too specific for a given context. For example, in #11 it is stated that the approach is best suited for contexts in which common customer needs are being addressed in similar application domains.

Table 3.7 – Benefits in using KM in HCI design context.

Benefits	Approaches	Total
Enable replicability of domain/context knowledge	#03, #06, #07, #09, #12	5
Improve product quality	#02, #05, #06, #12	4
Improve communication	#01, #03, #11	3
Increase team engagement/empowerment	#02, #06	2
Increase organizational integration	#03, #08	2
Reduce design effort	#03, #12	2
Improve design conceptualization	#03, #07	2
Promote standardization	#02	1
Increase productivity	#11	1
Promote organizational competitive advantage	#02	1
Decrease implementation and maintenance effort	#08	1
Decrease implementation and maintenance costs	#08	1

Table 3.8 – Difficulties in using KM in HCI design context.

Difficulties	Approaches	Total
Knowledge is often context-specific	#02, #06, #09, #11	4
Issues related to features of the KM technologies	#05, #06, #09	3
Low team engagement/empowerment	#01, #05, #08	3
User involvement	#07, #12	2
Integration of the KM approach into the organization	#06, #11	2
KM implementation and maintenance effort	#08, #09	2
Lack of consensus about HCI design conceptualization	#01, #02	2

3.4 Discussion

Taking the period of publications into account (RQ1), we can notice a long-term effort regarding the use of KM in HCI design, since this topic has been targeted by researchers for more than 20 years. However, the low average of publications per year (0.6 since 1995) shows that the topic has not been widely addressed. We can also notice that most of the publications are from the 2000s decade. The low percentage of journal publications, which generally require more mature works, can be seen as a reinforcement that the research on this topic is not mature enough yet. Besides, results about the research type (RQ2) show that only 40% of the works included some kind of evaluation, being only 13% evaluation of solutions in practice. This can be a sign of difficulty in applying the proposed approaches in industry, what reinforces that research on this topic is not mature enough yet and there seems to be a gap between theory and practice.

Concerning RQ3, we can notice that using KM in HCI design has been motivated mainly by delivering better products to users or optimizing the HCI design process in terms of effort, time and cost. Improving performance of the HCI design team was also mentioned, what is consistent with the other motivations related to the HCI design process, since increasing performance can contribute to decrease effort, time and cost. By analyzing the

results of approaches that applied some validation or evaluation, we noticed that only two (#03 and #12) provided results related to the initial motivation for using KM in HCI design (reduce design effort and improve product quality, respectively). The other publications were more focused on validating or evaluating features or functionalities of the proposed solutions. A common concern in several publications was the need for HCI design expert consultants, which can increase HCI design cost and effort. Capturing and reusing knowledge contribute to retain organizational knowledge and reduce dependence on external consultants. Another concern refers to communication problems. Smith and Dunckley (2002) highlight that barriers to effective communication between designers, HCI specialists and users, due to their differing perspectives, affect product quality. KM solutions are helpful in this context.

Usability has been the focus of the KM initiatives in HCI context (RQ4). In fact, this is not a surprise, because usability has been one of the most explored HCI aspects in the last years. Moreover, this property is quite comprehensive and includes other important aspects of HCI design, such as learnability, memorability, efficiency, safety and satisfaction (ISO, 2019). However, there are other important properties not addressed in the analyzed papers, such as user experience, communicability and accessibility. The knowledge items managed by the KM approaches are quite diverse. Design solutions, guidelines, test results and design patterns are some knowledge items found in different publications. Despite the variety of knowledge items, we noticed that most of the approaches (66.7%) manage up two different knowledge items. By analyzing the coverage of the approach in terms of single or multiple projects, we found out that four approaches (#01, #03, #07 and #12) manage knowledge involved in a single project, while the other eight approaches are more extensive, accumulating knowledge from multiple projects. In order to elevate knowledge reuse to the organizational level, it is important that a KM approach comprehends multiple projects in that organization.

Concerning knowledge use and capture (RQ5), at first, we expected that knowledge was captured and used in the same activity of the HCI design process. Therefore, results showed us that the same knowledge can be produced and consumed in different parts of HCI design process. For example, there are more approaches capturing knowledge in design evaluation activity than using in it. This reinforces the iterative characteristic of HCI design, where knowledge obtained in evaluation activity in one cycle can be used to improve the design in the next cycle.

Different technologies have been used to implement KM in HCI design context (RQ6). The most common are system-based approaches that use software to support KM process and store knowledge. We expected this result because KM systems, knowledge-based systems and knowledge repositories are widely adopted technologies in KM area. On the other hand, only two approaches use specific HCI techniques, namely contextual inquiry and heuristic evaluation. This may indicate that KM traditional approaches are suitable for addressing KM problems in HCI design (what was indeed expected) and that HCI techniques can be used to address specificities of the HCI design domain. Earlier steps of the development of KM solutions, such as knowledge analysis and modeling, are also addressed in some publications. Moreover, there is also concern with later steps, like the integration of the KM system into the organization. Some approaches combine different technologies, what can be a sign that the use of different techniques is a good strategy to address a more complete KM approach in HCI design.

As for benefits and challenges of using KM in HCI design context (RQ7), when categorizing the findings, we noticed that several of them are benefits and challenges of using KM in general. However, by analyzing the context of each KM approach, we can better understand how the findings relate to HCI design. For example, regarding the benefit *improve communication*, the works highlight the use of KM to support communication among the different actors involved in the HCI design process. In #10, communication between HCI specialists, designers and users is mediated by prototypes aiming at an agreement about the system design. In #01, KM facilitates the elicitation of the user's knowledge for the designer to apply it into the design. In #03, KM reduces errors of interpretation and contextualization among the people involved in the system design.

Some of the identified challenges and benefits are opposite each other. For example, on one hand, there is the challenge *low team engagement*. On the other hand, the benefit *increase team engagement*. We kept both because they were cited in different publications, thus under different perspectives. Moreover, we can see the challenge as a difficulty that, when overcome by the use of KM, can be turned into a benefit.

By analyzing the most cited benefit and challenge, we noticed that the generality level of the knowledge is an important question in a KM approach. The most cited benefit points to knowledge replicability in a specific context/domain. The most cited challenge points to the fact that it is difficult to generalize knowledge. Looking at data from RQ5, we noticed that approaches that reported knowledge generalization challenge handle knowledge from multiple projects, while approaches handling knowledge in a single project reported easy

replication of knowledge. Thus, how general will be knowledge should be determined by the context where the KM approach will be applied. When dealing with high diversity of knowledge and contexts, it becomes harder to produce general knowledge to be widely used to solve specific problems and be adopted in different contexts. One way of achieving improvements in replicability is using knowledge-based analysis methods, as reported by the approaches #03 and #07.

Based on the panorama provided by the mapping study results, in summary, we can say that KM has not been much explored in HCI context; it has been used mainly to improve software quality and HCI design process efficiency; it has focused on usability; and the KM approaches have been based on systems and repositories. As for benefits, KM has enabled knowledge replicability, improved product quality and communication. The main difficulties have been to generalize knowledge, address issues related to features of the system and low engagement of the team.

3.5 Threats to validity

As any study, our mapping study has some limitations that must be considered together with the results. Following the classification presented by (PETERSEN; VAKKALANKA; KUZNIARZ, 2015), next we discuss the main threats to the mapping study results.

Descriptive Validity is the extent to which observations are described accurately and objectively. To reduce descriptive validity threats, a data collection form was designed to support data extraction and recording. The form objectified the data collection procedure and could always be revisited. However, data extraction and recording still involved some subjectivity and was dependent on the researcher decisions. An important limitation in this sense is related to the classifications we made. We defined classification schemas for categorizing data in some research questions. Some categories were based on classifications previously proposed in the literature (e.g., type of research (WIERINGA et al., 2005)). Others were established during data extraction, based on data provided by the analyzed publications (e.g., RQ4). With an aim towards minimizing the threat, data extraction, classification schemas and data categorization were done by the first and second authors and reviewed by the other two authors. Discordances were discussed and resolved. However, determining the categories and how data fit them involves a lot of judgment. Thus, different results could be obtained by other researchers.

Theoretical Validity is determined by the researcher's ability to capture what is intended to be captured. In this context, one threat refers to the sources. We used four digital libraries selected based on other secondary studies in Software Engineering. Although this set of digital libraries represents a comprehensive source of publications, the exclusion of other sources may have left some valuable publications out of our analysis. ACM was not included in the sources because Scopus covers most of its publications. However, there are HCI publications indexed by ACM and not indexed by Scopus, which may have jeopardized the mapping results. To minimize this risk, we performed snowballing. Another threat refers to the fact that the study focused on scientific literature and did not include other alternatives, such as grey literature, that could enhance the systematic mapping coverage. Hence, extending this study with a multivocal literature review through grey literature analysis could complement and enrich the obtained results.

There are also limitations related to the adopted search string. Even though we have used several terms, there are still synonyms that we did not use. For example, since KM is a subjective area, many publications may have addressed KM aspects using other words such as "collaboration" and "organizational learning", which were not covered by our search string. Moreover, we did not include HCI and KM acronyms alone (HCI was combined with "design"), which could be an additional threat. However, the string includes the full terms referring to HCI and KM and we believe that it is probable that publications including the acronyms also include the full terms in either their title, abstract or keywords. Hence, our search string might have covered them anyway.

The researcher bias over publications selection, data extraction and classification is also a threat to theoretical validity. To minimize this threat, as we previously said, the steps were initially performed by the first and second authors and, to reduce subjectivity, the other two authors performed these same steps. Discordances and possible biases were discussed until reaching a consensus.

Finally, *Interpretive Validity* is achieved when the conclusions drawn are reasonable given the data obtained. The main threat in this context is the researcher bias over data interpretation. To minimize this threat, like in the other steps, interpretation was performed by the first and second authors and reviewed by the other two. Discussions were carried out until a consensus was reached. However, subjectivity still relies on the qualitative interpretation and analysis.

Even though we have treated many of the identified threats, the adopted treatments involved human judgment, therefore the threats cannot be eliminated and must be considered together with the study results.

3.6 Concluding Remarks

In this chapter, we presented a mapping study that investigated the state of the art concerning the use of KM in the HCI design context. The results of the mapping study provide a panorama of research related to the topic. We noticed that, although HCI design is a favorable area to apply KM, there have been only few publications exploring this research topic.

In order to complement the mapping study and provide an overview of KM in HCI design and insights of how to better apply KM to aid in HCI design practice, we carried out a survey with HCI design practitioners, aiming to investigate KM in HCI design practice. The survey is addressed in the next chapter.

Chapter 4

Survey: KM in HCI Design Practice

This chapter presents a survey carried out with HCI design practitioners aiming to investigate knowledge management in HCI design practice. It is organized as follows: Section 4.1 presents the chapter introduction; Section 4.2 addresses the survey planning and execution; Section 4.3 summarizes the obtained results; Section 4.4 discusses the results; Section 4.5 presents some of the limitations of the survey; Section 4.6 presents a consolidated view of the findings of the systematic mapping addressed in Chapter 3 and the survey; and Section 4.7 presents the chapter concluding remarks.

4.1 Introduction

The systematic mapping provided information about KM approaches to support HCI design according to the literature records. After conducting the mapping study, we performed a survey with 39 Brazilian HCI design practitioners to investigate KM in HCI design practice.

A survey is an experimental investigation method usually done after the use of some technique or tool has already taken place (PFLEEGER, 1994). Surveys are retrospective, i.e., they allow to capture an “instant snapshot” of a situation. Questionnaires and interviews are the main instruments used to apply a survey, collecting data from a representative sample of the population. The resulting data are analyzed, aiming to draw conclusions that can be generalized for the whole population represented by that sample (MAFRA; TRAVASSOS, 2006). In this work, we intended to reach many participants and analyze data objectively and quantitatively. Thus, in our survey, we decided to use a questionnaire containing objective questions.

We followed the process defined in (WOHLIN et al., 2012) which comprises five activities. *Scoping* is the first step, where we scope the study problem and establish its goals. *Planning* comes next, where the study design is determined, the instrumentation is considered and the threats to the study conduction are evaluated. *Operation* follows from the design, consisting in collecting data which then are analyzed and evaluated in *Analysis and Interpretation*. Finally, in *Presentation and Package*, the results are communicated.

4.2 Survey Planning and Execution

The study *goal* was to investigate aspects related to KM in HCI design practice. Aligned to this goal, we defined the *research questions* presented on Table 4.1, which were based on the systematic mapping research questions and results.

Table 4.1 – Survey: research questions and their rationale.

ID	Research Question	Rationale
RQ1	<i>Which stakeholders have been involved in HCI design practice?</i>	Identify which stakeholders have been involved in HCI design practice, which helps identify different perspectives and information needs in HCI design.
RQ2	<i>Which knowledge have been involved in HCI design practice?</i>	Investigate which knowledge has been involved in HCI design practice, particularly knowledge items (e.g., design solutions, guidelines and lessons learned) and design artifacts (e.g., wireframes, mockups and prototypes) used as sources of knowledge or produced to record useful knowledge.
RQ3	<i>Which HCI design activities have demanded better KM support?</i>	Investigate which HCI design activities have needed better support of KM (e.g., because there have not been enough knowledge resources to support their execution).
RQ4	<i>How has KM been applied in HCI design practice?</i>	Investigate how KM principles have been applied and identify technologies (e.g., tools, methods, etc.) that have been used to support knowledge access and storage in HCI design practice.
RQ5	<i>Which benefits and difficulties have been noticed when using KM in HCI design practice?</i>	Identify benefits and difficulties that have been experienced by practitioners when applying KM in HCI design practice and verify if practitioners have experienced more benefits or difficulties.
RQ6	<i>Which goals the use of KM in HCI design practice has contributed to achieve?</i>	Identify to which goals the use of KM in HCI design has contributed, aiming to figure out predominant reasons for using KM in HCI design practice.

The *participants* were 39 Brazilian professionals with experience in HCI design of interactive software systems. The participants profile was identified through questions regarding their current job positions, education level, knowledge of HCI design and practical experience in HCI design activities. Most participants (79.5%) declared to play roles devoted to HCI design activities (nine UX/UI designers; six UX designers; four product designers, two designers, two UX research designers, one art director, one IT analyst & UX designer, one interaction designer, one lead designer, one lead UI designer, one staff product designer and one UI designer). Others (20.5%) play roles that perform some activities related to HCI design (one programmer, one requirement analyst, one chief growth officer, one product owner, one IT analyst, one IT manager, one marketing manager and one project leader). Although these roles cannot be considered HCI design experts, we did not exclude these participants because they declared to have practical experience and knowledge in HCI design (probably acquired in their previous job and academic experiences). Moreover, even playing roles not dedicated to HCI design, they are often involved in HCI design in some way. Eight

participants (20.5%) had masters' degree, 26 (66.7%) had bachelor's degree, and five (12.8%) had not yet finished bachelor's degree course.

All participants declared theoretical knowledge of HCI design. Four of them (10.3%) declared low knowledge (i.e., knowledge acquired by himself/herself through books, videos or other materials). 16 participants (41%) declared medium knowledge, acquired mainly during courses or undergraduate research. Finally, 19 participants (48.7%) declared high knowledge (i.e., they are experts or have a certification, Masters or Ph.D. degree related to HCI design). Some areas of the courses cited by participants that declared medium or high knowledge are Design (46.2%), Computer Science (38.5%), Arts (28.2%), Social Communication (15.4%) and User Experience (7.7%). The participants were allowed to choose more than one option, hence the sum of the values is over 100%. Other areas such as Anthropology, Neuroscience, Information Science, Psychology were also mentioned by one participant each. 26 participants (66.7%) declared more than three years of experience in HCI design practice, 11 participants (28.2%) declared between one and three years and two (5.1%) declared less than one year.

The *instrument* used in the study consisted of a questionnaire composed of 10 objective questions. Most answer options of each question were defined based on the mapping study results. For example, when asked about the goals achieved with the help of KM in HCI design (RQ6), the options provided to the participants refer to the goals we found in the mapping study. However, some options were rewritten in a way that could enhance participants understanding (e.g., we changed "test results" to "previous design evaluation results" on RQ2) and others were added based on the authors' knowledge and experience (e.g., we included forums, blogs and social networks in RQ4). Furthermore, most questions also allowed the participant to provide additional information in text boxes to complement his/her answers. For example, besides selecting goals from the list provided in the question related to RQ6, the participants were also allowed to include new goals in their answers. The questionnaire is available at <http://bit.ly/Questionnaire-KM-in-HCI-design>.

The *procedure* adopted in the study consisted in sending the invitation to participate in the study, receiving the answers, verifying them, consolidating and analyzing data. The invitation was posted in discussion groups on Facebook, LinkedIn and Interaction Design Foundation's website². The researchers also sent the invitation by email to potential

² <https://www.interaction-design.org>

participants. Since the platforms did not inform how many people visualized the posts, we could not infer the percentage of invites that led to answers.

Before sending the invitation, we performed a pilot with three participants. Considering the participants' feedback, we improved the questionnaire aiming to ensure that the questions were clear and understandable. The invitation to participate in the study was posted on social medias and sent by email on December 16th, 2020. We received answers until January 11th, 2021. We received 40 answers to the questionnaire, however, after analyzing the participants profile related to HCI design knowledge and experience, we excluded one participant who reported to have low knowledge and experience with HCI design and did not answer some of the questionnaire questions. After that, each provided answer was verified and data was consolidated and analyzed against the research questions.

4.3 Results

In this section, we present a synthesis of the survey data for each research question presented on Table 4.1.

Stakeholders involved in HCI design practice (RQ1): aiming to identify stakeholders involved in HCI design practice, we asked the participants to identify the stakeholders they directly interact with in their HCI design practice. As it can be seen in Table 4.2, *developer* has been the most common stakeholder involved in HCI design practice, being mentioned by 37 participants (94.9%). Following that, *project manager*, *designer*, *user* and *client* were mentioned, respectively, by 34 (87.2%), 33 (84.6%), 27 (69.2%) and 26 (66.7%) participants. *Product owner* was cited by three participants (7.7%) and others (*business analyst*, *customer experience analyst*, *data analyst*, *HR people*, *product manager* and *scrum master*) were mentioned only once.

Table 4.2 – Stakeholders involved in HCI design practice.

Stakeholder	Number of participants	%
Developer	37	94.9%
Designer	34	87.2%
Project Manager	33	84.6%
Client	27	69.2%
User	26	66.7%
Product Owner	3	7.7%
Business Analyst	1	2.6%
Customer Experience Analyst	1	2.6%
Data Analyst	1	2.6%
HR People	1	2.6%
Product Manager	1	2.6%
Scrum Master	1	2.6%

Knowledge involved in HCI design practice (RQ2): first, the participants were asked about the *knowledge items* they use or produce during HCI design activities. We consider as knowledge items pieces of knowledge that can be useful in HCI design, such as lessons learned, standards, guidelines and patterns. Figure 4.1 presents the results of this question. Some items have been used and produced by a high number of participants: *organizational design standards* (used by 34 participants, 87.2%, and produced by 26 participants, 66.7%), *lessons learned* (used by 34 participants, 87.2%, and produced by 24 participants, 61.5%), *guidelines* (used by 34 participants, 87.2%, and produced by 22 participants, 56.4%) and *libraries of design components or elements* (used by 32 participants, 82.1%, and produced by 23 participants, 59%). Other knowledge items have also been used by many participants, but produced by a smaller number, such as *examples* (used by 34 participants, 87.2%, and produced by 14 participants, 35.9%), *design solutions from the organization* (used by 35 participants, 89.7%, and produced by 18 participants, 46.2%) and *design solutions from outside the organization* (used by 35 participants, 89.7%, and produced by 11 participants, 28.2%). In general, HCI design practitioners have used and produced different knowledge items (11.1 and 6.6 in average, respectively).

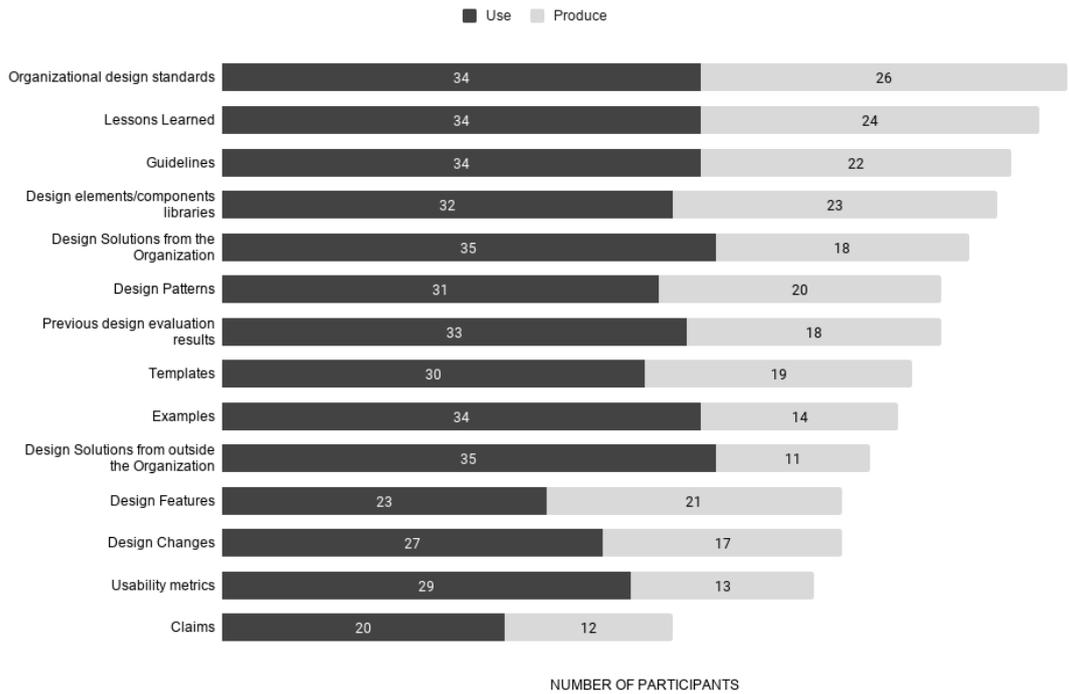


Figure 4.1 – Knowledge items used and produced in HCI design practice.

The participants were also asked about design artifacts they use or produce during HCI design activities. We use the term design artifact to refer to documents, models, prototypes and others that record information about the design solution. Figure 4.2 shows the results. *User requirements*, *scenarios* and *interaction models* were the most cited as artifacts used during HCI design. On the other hand, *wireframes*, *functional prototypes* and *mockups* were the most cited as artifacts produced during HCI design.

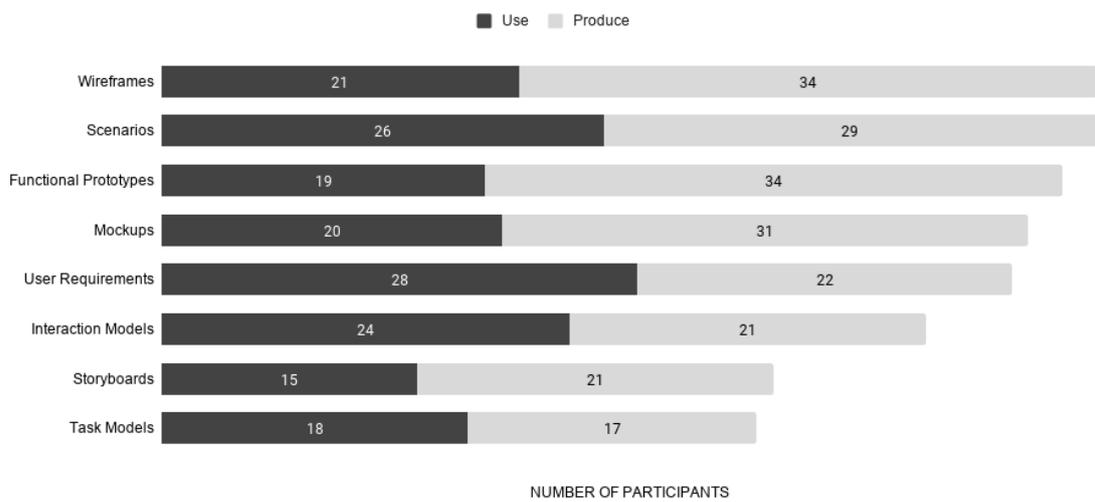


Figure 4.2 – Design artifacts used and produced in HCI design practice.

We also asked the participants to inform whether the artifacts used and produced by them sufficiently provide all information needed to describe the HCI design solution (i.e., if the knowledge recorded in the artifacts is enough to the implementation and evaluation of the solution). 26 participants (66.7%) answered “yes” and 13 (33.3%) answered “no”. Eight out of the 13 participants pointed out they miss information about personas, user research data and usability tests. These 13 participants were also asked about the ways the missing information is communicated. The results are presented in Table 4.3. *Annotations* and *talks* have been the most used ways (eight participants, 61.5%) to complement information provided in design artifacts. Seven participants (53.9%) reported the use of *meetings*, while one uses *documentation* or *specific tools*. The participants indicated that annotations and talks have been used informally, while meetings, documentation or tools have been used systematically, following organizational practices.

Table 4.3 – Ways to obtain missing information.

Method	Number of participants	%
Annotations	8	61.5%
Talks	8	61.5%
Meetings	7	53.9%
Documentation or Tool	1	7.7%
None	1	7.7%

HCI design activities demanding better KM support (RQ3): taking the HCI design activities established by ISO 9241-210 (ISO, 2019) as a reference, the participants were asked to judge whether the knowledge resources (e.g., knowledge items, artifacts) used by them have provided sufficient knowledge to support each activity. Figure 4.3 presents the results. In general, most participants consider that they have access to enough knowledge to perform HCI design activities. *Produce design solutions* has the highest number of participants (31 participants, 79.5%) reporting to have had sufficient knowledge to perform it. On the other hand, *evaluate design solutions* has the highest number of participants (10 participants, 25.6%) declaring that the available knowledge has not been enough. Sixteen participants (41%) declared to have not had sufficient knowledge to support at least one HCI design activity. They pointed out that, in order to address the lack of knowledge, they have performed user research, searched for successful use cases, talked to stakeholders, and looked at the literature.



Figure 4.3 – Available knowledge to support HCI design activities.

How KM has been applied in HCI design practice (RQ4): Figure 4.4 shows the approaches that have been used to support knowledge access or storage in HCI design practice. *Brainstorming* and *blogs* have been the most used ways to access knowledge (28 participants, 71.8%), followed by *mental models* and *electronic documents and spreadsheets* (26 participants, 66.7%). Except by *blogs*, those have also been the most used ways to store knowledge: *brainstorming* has been used by 27 participants (69.2%); *mental models and electronic documents and spreadsheets* by 24 (61.6%). *Ontologies* have been the less used way by the participants. Only 7 participants (18%) have used ontologies to access knowledge and 5 participants (12.8%) have used it to store knowledge. Concerning knowledge storage, *social networks* (6 participants, 15.4%) and *forums* (8 participants, 20.5%) have also not been much used. In general, the approaches shown in Figure 4.4 have been more used to support knowledge access than to support knowledge storage.

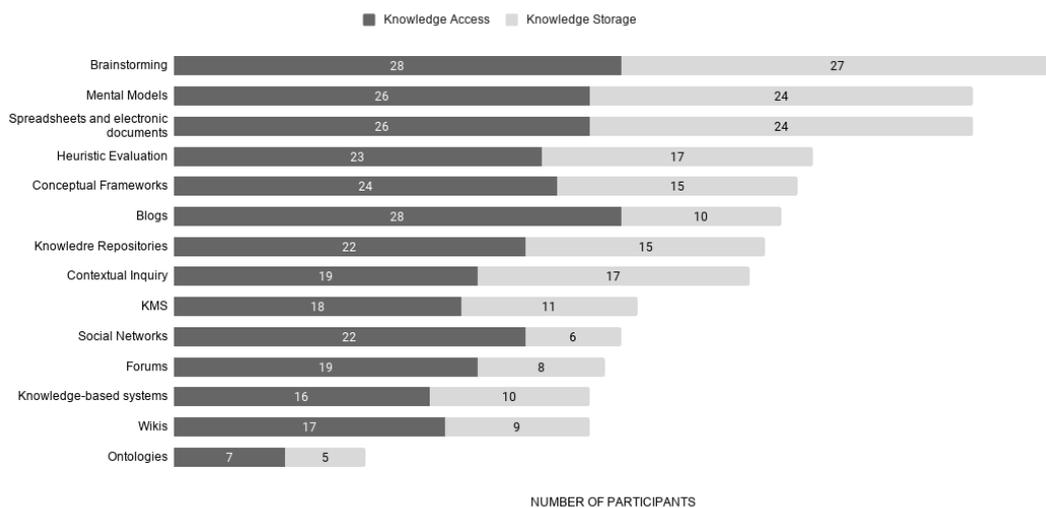


Figure 4.4 – Approaches to support knowledge access and storage in HCI design.

Benefits and difficulties of using KM in HCI design practice (RQ5): 34 participants (87.2%) reported to perform KM practices to support HCI design activities. 16 of them (41.0%) have followed institutionalized organizational practices, while 18 (46.2%) have performed on their own initiative. These 34 participants were asked about benefits and difficulties they have perceived in using KM to support HCI design. The results are summarized on Table 4.4 and Table 4.5.

Table 4.4 – Benefits of using KM in HCI design practice.

Benefit	Number of participants	%
Enable replicability of domain or context knowledge	27	79.4%
Promote standardization	26	76.5%
Improve communication	25	73.5%
Increase productivity	24	70.6%
Reduce design effort	24	70.6%
Improve product quality	23	67.6%
Improve design conceptualization	20	58.8%
Improve team learning	18	52.9%
Reduce dependency on specialists	18	52.9%
Increase team engagement or empowerment	17	50.0%
Increase organizational integration	16	47.1%
Reduce design cost	16	47.1%
Promote organizational competitive advantage	11	32.4%

Table 4.5 – Difficulties of using KM in HCI design practice.

Difficulty	Number of participants	%
Low team engagement or empowerment	16	47.1%
KM implementation and maintenance effort	15	44.1%
Integration of the KM approach into the organization	15	44.1%
Lack of consensus about HCI design conceptualization	14	41.1%
Find relevant knowledge to a given context	13	38.2%
Low user involvement	9	26.5%
Issues related to features of the KM technologies	8	23.5%
Unclear business model	1	2.9%

Goals to which the use of KM in HCI design practice has contributed (RQ6): Aiming to identify the predominant reasons for using KM in HCI design practice, the participants were asked how much KM support to HCI design contributes to achieve certain goals. The goals presented to them were identified in the systematic mapping as motivations to perform KM in HCI design context. Figure 4.5 shows the results.

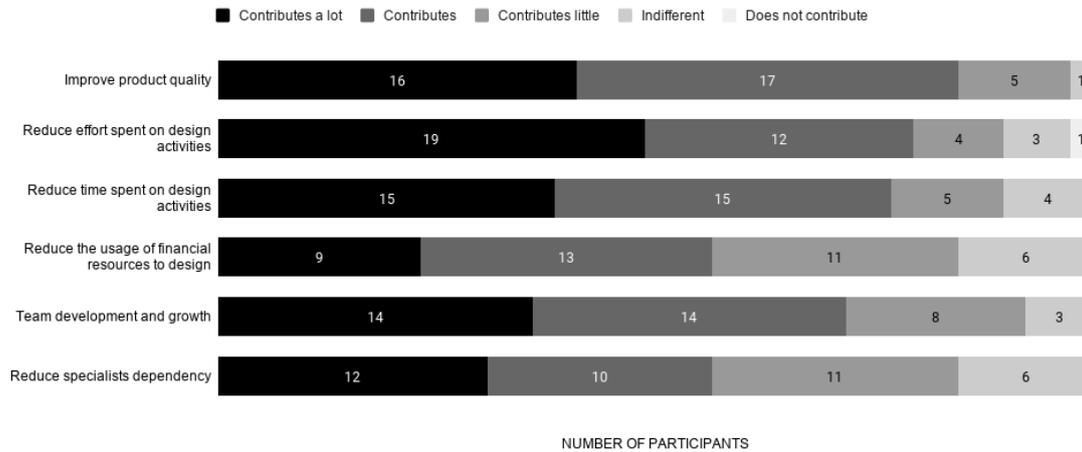


Figure 4.5 – KM contribution to goals achievement when supporting HCI design.

According to the participants, the goals to which using KM in HCI design contributes the most are *improve product quality* (84.6% of the participants stated that KM contributes a lot or contributes to it) and *reduce effort spent on design activities* (79.5% of the participants stated that KM contributes a lot or contributes to it). On the other hand, the participants have seen less contribution of KM in HCI design to *reduce the usage of financial resources in design* and to *reduce the dependency on specialists* (43.6% of the participants stated that KM contributes little or is indifferent to both of them).

4.4 Discussion

In this section, we present some discussion about the results shown in the previous section.

By analyzing the participants profile, we noticed that several stakeholders (20.5%) who had knowledge of and experience with HCI design did not play a role devoted to HCI design by the time of the survey execution. We believe that this reinforces the multidisciplinary nature of HCI design and corroborates with a recent finding from (NETO et al., 2020) that some professionals may choose to pursue a double background involving design and development areas.

Concerning *stakeholders* (RQ1), it can be noticed a variety of them being involved in HCI design. Considering that the interactions usually occur in the context of projects, the results indicate that teams of HCI design projects have included designers, developers, project managers, and frequently also have involved clients and users. These stakeholders have different roles in HCI design, and thus may have different HCI design knowledge needs. For example, a developer may need to implement the design solution presented in a design artifact. For that, this artifact should present technical decisions that affect the

implementation. A project manager, in turn, may need to have a broader view of several design artifacts to verify if the implemented solution satisfies the requirements agreed with the client. Hence, it is important that KM approaches consider needs of different stakeholders to properly support HCI design. Moreover, it may be necessary to integrate knowledge from different sources to provide a solution that integrates needs of different stakeholders. This can be done, for example, with a knowledge management system with multiple views for each different role.

Regarding *knowledge involved in HCI design* (RQ2), by analyzing the knowledge items used and produced in HCI design practice, we can notice which knowledge has been more useful to practitioners. Most participants use knowledge items that provide design knowledge obtained from previous design experiences, such as design solutions from the organization, design solutions from outside the organization and examples. This can be a sign that new designs have been created based on previous experiences adapted to the new context. However, these knowledge items have not been much produced by the participants. This may be due to the effort required to record knowledge for future reuse. Hence, it would be important to facilitate capture, recording and retrieval of knowledge embedded in design solutions. On the other hand, two of the knowledge items produced by the highest number of participants (organizational design standards and guidelines) record general principles and practices to be followed when designing HCI solutions. This may indicate that the participants have found it easier to produce knowledge independent of specific solutions. Considering the relation between the number of knowledge items used and produced by the participants, the higher number of used items shows that, in general, the participants have acted more as knowledge consumers than knowledge producers. This may happen because either the participants do not have enough time to produce knowledge items, or the knowledge production is done by someone else. Consulting knowledge directly helps designers in the activities they were doing in that moment. In contrast, knowledge production does not seem to be immediately useful to them, although it is important in an organizational level. We believe that approaches that promote knowledge recording and storage requiring less effort could motivate designers to act as knowledge producers.

As for design artifacts, we noticed that the ones produced by more participants (wireframes, functional prototypes and mockups) represent abstractions of the design solution. Hence, the creation of such artifacts is part of the design solution development. On the other hand, the artifacts used by more participants (user requirements, sceneries and interaction models) provide useful information to develop the design solution (i.e., they

represent inputs to design development). One third of the participants (33.3%) considered the artifacts used or produced by them limited to meet information needs about the design solution and reported the use of complementary ways to transfer missing knowledge. When analyzing the three most cited ways, we observed that two of them (talks and meetings) are based on conversation between team members. This can be a sign that it may be difficult to articulate certain knowledges in artifacts. This is reinforced by the high usage of annotations, which are less formal and structured, and the low usage of documentations and tools. Besides, considering that the use of more than one method of knowledge transfer is a common practice used by the participants, it is likely that they prefer to have this communication redundancy as a way of reinforcing the understanding of all stakeholders about the design. Therefore, we believe that the missing knowledge in HCI design artifacts can be transferred, for example, by performing regular meetings and by providing means to easily attach additional annotations on design artifacts.

Concerning *HCI design activities* (RQ3), ‘produce design solutions’ was the one in which the largest number of participants (79.5%) indicated to have enough knowledge access. This can be a sign that participants have used knowledge mainly to support the creation of design solutions. On the other hand, a high number of participants indicated not having sufficient knowledge to perform the activities ‘understand and specify the context of use’ (23%), ‘specify user requirements’ (23%) and ‘evaluate the design solution’ (25.6%). Therefore, it is necessary to identify useful knowledge to support these activities (e.g., missing knowledge related to personas and user research data, as reported in RQ2) and provide means to represent and access it in an easy way.

As for the *approaches to support knowledge access and storage in HCI design* (RQ4), it can be observed that the most used approaches, such as brainstorming, mental models and electronic spreadsheets and documents, usually support both knowledge access and storage. This may suggest that it is easier and simpler to implement and use them. Brainstorming for example, has the advantage of the participants share and obtain knowledge at the same time. On the other hand, web-based resources, such as blogs, forums and social networks are more used to support knowledge access than knowledge storage. Probably, these resources have been used more as sources of inspiration to bring new ideas from outside the organization. In addition, the reason why these resources have been less used by practitioners to record knowledge may be a concern in not exposing organizational design knowledge on the internet. HCI design knowledge must be captured, recorded and propagated in order to be raised from the individual level to the organizational level. Hence, we believe that KM

initiatives in HCI design should consider approaches such the ones most used by practitioners to support both knowledge access and storage.

Concerning *benefits and difficulties of using KM in HCI design* (RQ5), most participants declared to have experienced KM practices in HCI design. 41.0% followed institutionalized practices and 46.2% have performed on their own initiative. This indicates that HCI design professionals have been concerned with the need of practices to help manage knowledge and are seeking for solutions by themselves when they are not provided by the organization. According to the participants, in general, using KM to support HCI design brings more benefits than difficulties. The most cited benefits were related to standardization, reuse, communication and productivity, while the most cited difficulties were related to the lack of consensus in HCI design conceptualization and to the effort of implementing, engaging the team and integrating the KM approach in the organization. Based on that, to effectively implement a KM approach, it would be interesting to convince people and the organization that the additional effort in the beginning is worth the benefits they obtain afterwards.

Finally, by analyzing *goals to which the use of KM in HCI design has contributed* (RQ6), 'reduce the usage of financial resources' and 'reduce the dependency on specialists' have been considered less impacted by the use of KM in HCI design. This may be because reducing costs can be a side effect of reducing time spent on design or producing better designs, with less errors. Moreover, even if expert's knowledge is transferred and managed at organizational level, user centered design deals with people, hence there are subjective aspects that still needs to be addressed by specialists. Another point to be considered is that the participants of the survey were, in the majority, HCI design experts, which could have biased their answers about the impact of using KM to reduce the dependency on HCI design experts. It is also important to note that 'reduce the effort spent on design activities' was the goal which participants believe to be most impacted by the use of KM in HCI design. By having in hand proper knowledge resources, the designer can learn from previous experiences, reuse solutions and explore more design alternatives, which can lead to designing better and more efficiently.

4.5 Threats to Validity

As discussed in the context of the systematic mapping, when carrying out a study, it is necessary to consider threats to the validity of its results. In this section we discuss some threats involved in the survey using the classification presented in (WOHLIN et al., 2012).

Internal Validity: It is defined as the ability of a new study to repeat the behavior of the current study with the same participants and objects. The main threat to internal validity is communication and sharing of information among participants. To address this threat, the questionnaire was made available online, so that the participants could answer it at the time they considered most appropriate. This can minimize the threat of communication, since participants were not physically close during the study and did not necessarily perform the study at the same time.

External Validity: It is related to the ability to repeat the same behavior with different groups of participants. In this sense, the limited number of participants and the fact that all of them are Brazilian professionals are also threats to the results. Moreover, some of the participants were invited based on the authors' relationship network, which may also have influenced the answers.

Construction Validity: It refers to the relationship between the study instruments, participants and the theory being tested. In this context, the main threat refers to the possibility of the participants have misunderstood some questions. To address this threat, we performed a pilot that allowed us to improve and clarify questions. Moreover, we provided definitions for the terms used and examples of information that should be included in the survey, so that the participants could better understand how to answer it.

Conclusion Validity: It measures the relationship between the treatments and the results and affects the ability of the study to generate conclusions. A threat to conclusion validity refers to the subjectivity in data analysis, which may reflect the authors' point of view. In addition, the results reflect the participants' personal experience, interpretation and beliefs. Hence, the answers can embed subjectivity that could not be captured through the questionnaire. These and the other threats discussed above affect the representativeness of the survey results and, thus, the results must be understood as preliminary evidence and should not be generalized.

4.6 Consolidated View of Findings

In this section, we present some discussions involving the systematic mapping and survey results, aiming to provide a consolidated view of the findings from both studies.

The three most cited motivations for using KM found in the systematic mapping (RQ3) are the same as the three goals most impacted by the use of KM in HCI design practice, according to survey participants (RQ6). This shows that, in general, it is expected

that the use of KM in HCI design can contribute to improve product quality and reduce effort and time spent on design activities.

Considering the most reported benefits and difficulties of using KM in HCI design, the survey results provided some ones not observed in the literature. For example, most survey participants reported ‘standardization’ and ‘productivity’ as benefits and ‘KM implementation and maintenance effort’ and ‘lack of consensus about HCI design conceptualization’ as difficulties. This difference is not a surprise, since the mapping results showed that most proposed approaches have not been applied in industry. We believe that to achieve success in implementing knowledge management, it is important to consider HCI design professionals’ perspective, pursuing the benefits and implementing strategies to overcome the difficulties.

There are other differences between the mapping and survey results. For example, traditional KM technologies, such as knowledge management systems, knowledge repositories and knowledge-based systems, have been the most used approaches reported in the literature, but have not been much used by HCI design professionals. The reasons why they do not use those approaches may be quite diverse, including being not aware that they exist or considering them too complex. Since 46.2% of the participants perform KM practices on their own initiative, it is likely that they have preferred simpler approaches that can be implemented by themselves. This reinforces the gap between industry and academy perceived from the analysis of the systematic mapping results. In order to decrease this gap, KM approaches to support HCI design should be closer to approaches that professionals are already familiar with, which can contribute to simpler and easier implementation and use.

Results from both studies show that design guidelines and design solutions have been reused in HCI design. Organizational design standards, lessons learned, and design component libraries have also been useful for HCI design professionals. Therefore, KM approaches to support HCI design should be able to handle these knowledge items, supporting their capture, storage and retrieval. As indicated by results from both studies, these knowledge items have probably been most used to support the activity ‘produce design solutions’. This was the activity in which most approaches found in the literature use knowledge and most participants considered having sufficient knowledge support. KM approaches should also provide support to other activities such as ‘understand and specify context of use’, ‘specify user requirements’ and ‘evaluate design solutions’, contributing to the HCI design process as a whole.

4.7 Concluding Remarks

This chapter and the previous one presented an investigation about the use of knowledge management in the HCI design context. To investigate the state of the art, we performed a systematic mapping presented in Chapter 3. After that, we carried out the survey presented in this Chapter, with 39 Brazilian professionals who work on HCI design. As the main result of the studies, we provided a panorama of research related to the topic and identified gaps and opportunities of improvements to organizations interested in applying KM initiatives in HCI design context.

For example: (i) The lack of a common conceptualization of HCI design (pointed out in #01 and #02 in the mapping study and also by 35.9% of the survey participants) leads to communication problems between the different actors involved in the HCI design process. We believe that the use of ontologies to establish this common conceptualization could help in this matter. However, since ontologies are not much familiar to practitioners (survey RQ4 results), ontology-based KM approaches in HCI design should abstract the ontology to final users (e.g., using the ontology to derive the conceptual model of a knowledge-based system). (ii) The gap between theory and practice (systematic mapping RQ2 results) shows that it is necessary to take KM solutions to practical HCI design environments. The survey results show that HCI design professionals are familiar with more robust KM approaches (such as knowledge management systems) but prefer to use simpler ways to deal with knowledge, such as brainstorming sessions and electronic spreadsheets and documents. Therefore, lightweight technologies and a divide and conquer strategy to reduce complexity of the conception, implementation and evaluation of a KM approach might be useful, allowing to provide results for the organizations in smaller periods of time and increasing benefits as the approach evolves. (iii) Other aspects besides usability (e.g., user experience, communicability and accessibility) should be explored in KM initiatives to improve HCI design. (iv) The benefits and difficulties identified in mapping (RQ7) and reported by the survey participants (RQ5) indicate issues that can be investigated in future research. For example, case studies can be carried out in organizations to evaluate the use of KM approaches in HCI design context.

We did not find any study investigating the use of KM in HCI design context. A work that can be related to ours is (STEPHANIDIS; AKOUMIANAKIS, 2001), consisting of a literature review about categories of computer aided HCI design tools and a proposal of a new category to address the knowledge complexity involved in HCI design. However, the

study focused on computational tools, not investigating how other kinds of KM approaches can help in HCI design process.

Chapter 5

Human-Computer Interaction Design

Ontology

This chapter presents the Human-Computer Interaction Design Ontology (HCIDO), the reference ontology about HCI design proposed in this work. Section 5.1 presents the chapter introduction. Section 5.2 presents the Software Design Reference Ontology (SDRO), which addresses design in the software context and was developed to be reused in HCIDO development. Section 5.3 presents HCIDO. Section 5.4 discusses related works. Finally, Section 5.5 presents the chapter concluding remarks.

5.1 Introduction

The investigation about KM in HCI design presented in chapters 3 and 4 indicated that the lack of a common conceptualization about HCI design has been one of the main challenges in applying KM to support HCI design of interactive systems. The use of ontologies can help to address this challenge, by providing a formal and explicit specification of a shared conceptualization (STUDER; BENJAMINS; FENSEL, 1998). Hence, we propose in this chapter the Human-Computer Interaction Design Ontology (HCIDO), which aims at establishing a common conceptualization of HCI design of interactive systems.

HCIDO is a networked domain ontology of HCI-ON (COSTA *et al.*, 2020) and reuses concepts related to HCI and SE by specializing other ontologies from HCI-ON and SEON (RUY *et al.*, 2016), respectively. From SEON, we highlight the Software Design Reference Ontology (SDRO), which was also developed in this work and addresses general design aspects in the software context, connecting them to other SE aspects, such as requirements, code and testing. HCIDO and SDRO are respectively placed in the domain layer of HCI-ON and SEON architectures, as it is shown in Figure 5.1 (the notation used in the figure is the same used in Figure 2.7 and explained in Section 2.3.3).

The development of HCIDO considered six of the seven requirements³ established during activities of the *Relevance Cycle* of the research method followed in this work, as we

³ (R1) the ontology must cover main aspects regarding HCI design, including not only the created artifacts but also mental aspects that precede the creation of design artifacts; (R2) the ontology must consider aspects related from both HCI and SE; (R3) the ontology must be modular; (R4) the ontology must be formally rigorous; (R5) the ontology must be ground in a well-founded ontology; (R6) the ontology must be developed by following an appropriate Ontology Engineering method; and (R7) the ontology must be used to solve problems.

explained in Chapter 1 (Section 1.4). The last requirement (R7) concerns HCIDO use. Hence, it was addressed after HCIDO development and will be discussed in Chapter 6. In order to meet the other six requirements, HCIDO is based on HCI design literature, standards and also in theories related to design in general (R1); it was developed as a networked ontology of HCI-ON (COSTA *et al.*, 2020) and reuses concepts from SEON ontologies (RUY *et al.*, 2016) (R2); it is organized into two sub-ontologies (R3); it is defined by means of conceptual models and textual descriptions (R4)⁴; it is grounded in UFO (GUIZZARDI, 2005) (R5) and (like SRDO) it was developed by following SABiO (FALBO, 2014), a Systematic Approach for Building Ontologies (R6).

SABiO supports the development of domain reference ontologies, as well as the design and coding of operational ontologies. SABiO was chosen because it has been successfully used to develop domain ontologies, in particular Software Engineering reference domain ontologies, including the ones already integrated to SEON and reused in this work, namely: Software Process Ontology – SPO (BRINGUENTE; FALBO; GUIZZARDI, 2011), System and Software Ontology – SysSwO (DUARTE *et al.*, 2018) and the Reference Software Requirements Ontology – RSRO (DUARTE *et al.*, 2018).

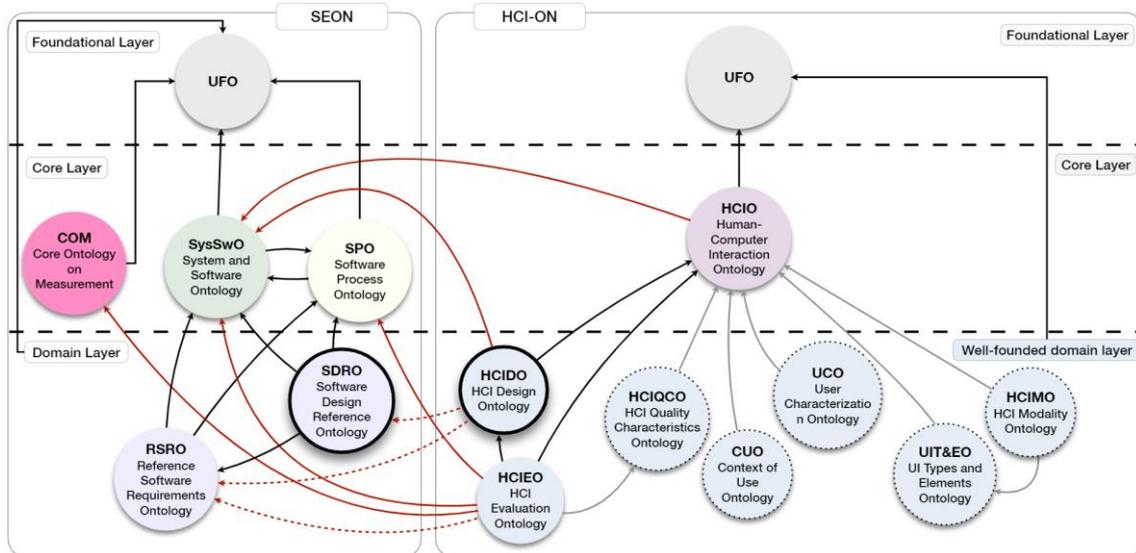


Figure 5.1 – Placement of HCIDO and SDRO in HCI-ON and SEON architectures.

SABiO’s development process comprises five main phases, namely (FALBO, 2014): (1) *Purpose Identification and Requirements Elicitation*; (2) *Ontology Capture and Formalization*; (3) *Design*; (4) *Implementation*; and (5) *Testing*. These phases are accompanied by support processes: *Knowledge Acquisition, Reuse, Documentation, Evaluation and Configuration Management*. SABiO aims

⁴ Due to time constraints to conclude this dissertation, axioms were not formally defined here and will be addressed in a future work.

at developing both reference ontologies (phases 1 and 2) and operational ontologies (phases 3, 4 and 5). SDRO and HCIDO are domain reference ontologies, thus only the first two phases were performed. During the *Purpose Identification and Requirements Elicitation* phase, we raised the competency questions that the ontologies should be able to answer. In *Ontology Capture and Formalization*, the required concepts were captured, formalized in UML diagrams and described. *Knowledge acquisition* was performed based on the literature and international standards, as well as by consulting domain specialists. *Reuse* consisted in reusing HCI and SE concepts from HCI-ON and SEON and general design concepts from the works by Ralph and Wand (2009) and Guarino (2014). *Evaluation* was conducted using verification (checking if the ontologies were able to answer the proposed competency questions) and validation (using data from real-world situations to instantiate the ontologies concepts). In *Documentation*, we produced the ontologies specifications presented in this chapter. Finally, during *Configuration Management*, we controlled different versions of the ontologies until reaching the version presented in this work. Details about the development of SDRO and HCIDO are presented in the next two sections.

5.2 Software Design Reference Ontology

Understanding the meaning of design in general helps understand what is design in the software development context, since the activity of designing shares many common characteristics across different fields (MCPHEE, 1996). The meaning of “design” in the dictionary can be either a verb (e.g., “to make or draw plans for something” (DESIGN, 2020a); “to conceive and plan out in the mind” (DESIGN, 2020b)) or a noun (e.g., “a drawing or set of drawings showing how a product is to be made and how it will work and look” (DESIGN, 2020a); “a mental project or scheme in which means to an end are laid down” (DESIGN, 2020b)). One meaning does not exclude the other, rather, they are complementary and suggest different viewpoints of the design phenomenon.

In the Software Engineering literature, we can also find definitions referring to “software design” as a verb and as a noun. For example, software design is defined as the process of describing architecture, components, modules, interfaces, and data for a software system, to specify how requirements are to be met by the implemented software, being applied regardless of the software process model that is used (BUDGEN, 2003; ISO/IEC/IEEE, 2017; PRESSMAN; MAXIM, 2020). The result of this process is a description that acts like a blueprint for constructing software, which is also referred as the software design.

A more formal conceptualization of “design” as a verb and as a noun was proposed by Ralph and Wand (2009), which was based on a literature review of design definitions across different fields, summarizing what they have in common and trying to resolve disagreements between them. Design as a noun is defined by these authors as “*a specification of an object (the design object), manifested by an agent, intended to accomplish goals, in a particular environment, using a set of primitive components, satisfying a set of requirements, subject to constraints*”, while as a verb, it is defined as the process of creating a design (RALPH; WAND, 2009). Hence, a designer is the agent who manifests a specification. A specification, in turn, is a detailed description of a design object’s structural properties and, it may be purely mental, presented as physical or symbolic representation or as the object itself. This is in accordance with the aforementioned definitions (from dictionary) and with the discussion provided by Guarino (2014). According to him, the design object is the thing being designed, which in the context of this work is software. It has essential characteristics that result from the design choices encoded in the design specification (GUARINO, 2014). These choices involve the selection and manipulation of components (or primitives) that will compose the designed object. Guarino and Melone (2015) highlighted that components that designers refer when designing have a different ontological status from the physical components that constitute the realized design object. They are called conventional system components and represent what designers have in mind. They exist in a particular place of the object, where they play a specific role. Goals, requirements, constraints and the design object environment are considered inputs to the design process (RALPH; WAND, 2009) and are all encompassed by term “requirements” in software development.

In this work, we propose the Software Design Reference Ontology (SDRO) to provide a formal conceptualization about the design phenomenon in the software context and to be reused in the development of HCIDO. SDRO is integrated to SEON, uses the works by Ralph and Wand (2009) and Guarino (2014) as a reference to describe the core design notions and reuses concepts from other SEON ontologies to address software particularities. By combining general notions of design and specific aspects of Software Engineering, it is possible to provide a conceptualization about design in the software context integrated to other aspects, such as requirements, code and testing. SDRO allows the instantiation of software design situations regardless of the design paradigm, process or method used in software development. It is focused on design as noun, i.e., it is not focused on describing activities involved in a general software design process, which is addressed in SEON by the Design Process Ontology (RUY *et al.*, 2016).

The ontology scope was defined by means of competency questions, i.e., questions the ontology must be able to answer and are used as a basis to develop the ontology conceptual model. Considering the ontology purpose, the following set of competency questions (CQ) was established:

Table 5.1 – SDRO Competency Questions

Competency Question	Rationale
CQ1. How does a software designer reason about the object being designed?	Understand the mental nature of software design.
CQ2. What is a software design specification? CQ3. Which are the components of a software design specification?	Understand what a software design specification is and its constituents.
CQ4. What is a software design object? CQ5. Which are the components of a software design object?	Represent the software built from a design specification and its components.
CQ6. What is described in a software design specification?	Understand the design-related information described in a design specification and the relation between the software design specification of an object and the mental elements of the design of that object.
CQ7. What is the motivation for a software design choice?	Understand the reason why a designer makes a design choice regarding a software design object.
CQ8. How can a software design object be implemented from a software design specification?	Understand how the information encoded in a software design specification is transformed into an implemented software design object.
CQ9. How can a software design object be evaluated against a software design specification?	Understand how the information encoded in a software design specification can be used to evaluate the implementation of the software design object.

SDRO addresses “design” as a noun, describing the mental and physical elements involved in the design of software systems and the relations between them. Here, the term “physical” is borrowed from other works (BAKER; HOEK, 2006; GUARINO, 2014; RALPH; WAND, 2009) referring to the perception of something through the senses. But it is important to highlight that software is something abstract. Thus, there are differences in the way it is perceived when compared to physical objects like a chair or a car. Considering these differences, we divided SDRO into two sub-ontologies: the *Mental Aspects* sub-ontology and the *Physical Aspects* sub-ontology. Figure 5.2 shows the organization of these sub-ontologies in SDRO and their dependency relations with other SEON ontologies. A dependency relation between two ontologies means that the source ontology reuses concepts from the target ontology.

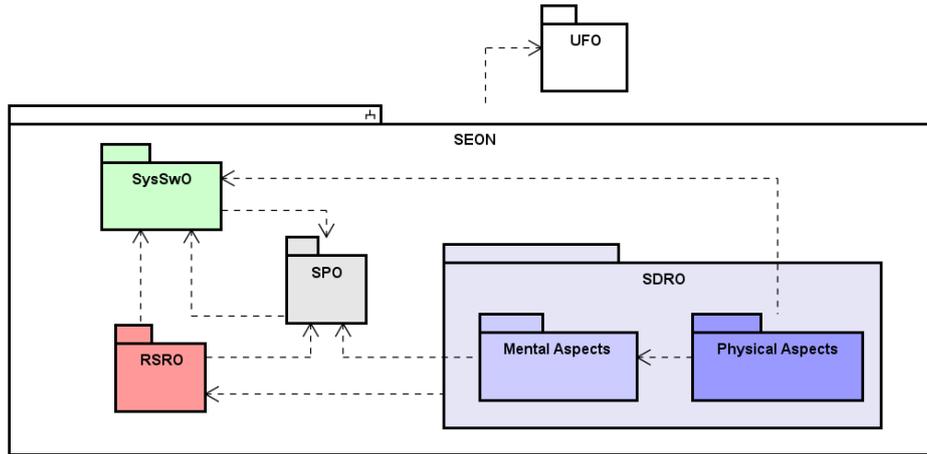


Figure 5.2 – SDRO architecture.

The *Mental Aspects* sub-ontology is presented in Figure 5.3, while the *Physical Aspects* sub-ontology is presented in Figure 5.4. In the figures, the black single-dashed horizontal lines separate concepts from different ontologies at the same layer. Red double-dashed lines separate the layers of SEON architecture. Different colors are used to indicate concepts from different ontologies (the same colors are used in Figure 2.7 for concepts from UFO, SPO, SysSwO and RSRO). Concepts from SDRO are presented in different shades of violet according to the sub-ontology to which they belong: light violet is used for *Mental Aspects* sub-ontology and dark violet for *Physical Aspects* sub-ontology. In the model description, SEON concepts are written in **bold** and SDRO concepts are written in ***bold italics***.

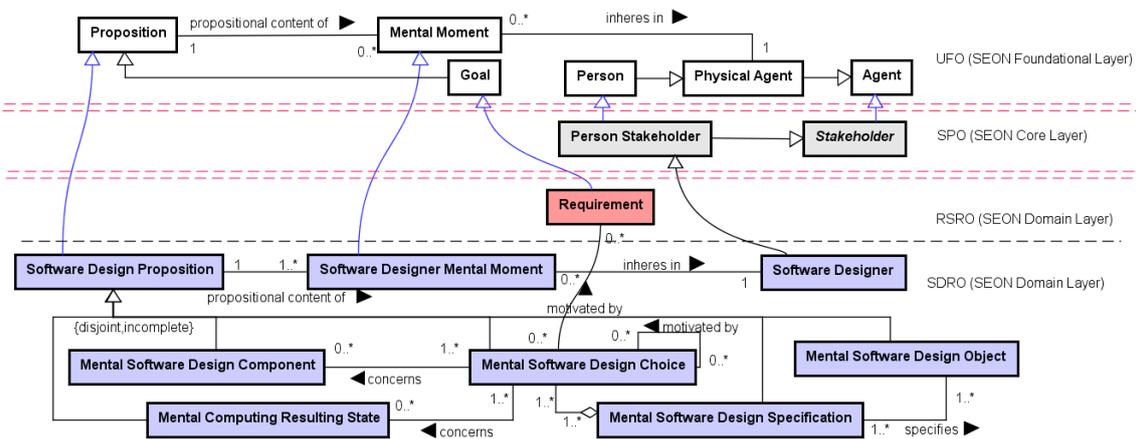


Figure 5.3 – SDRO Mental Aspects sub-ontology conceptual model.

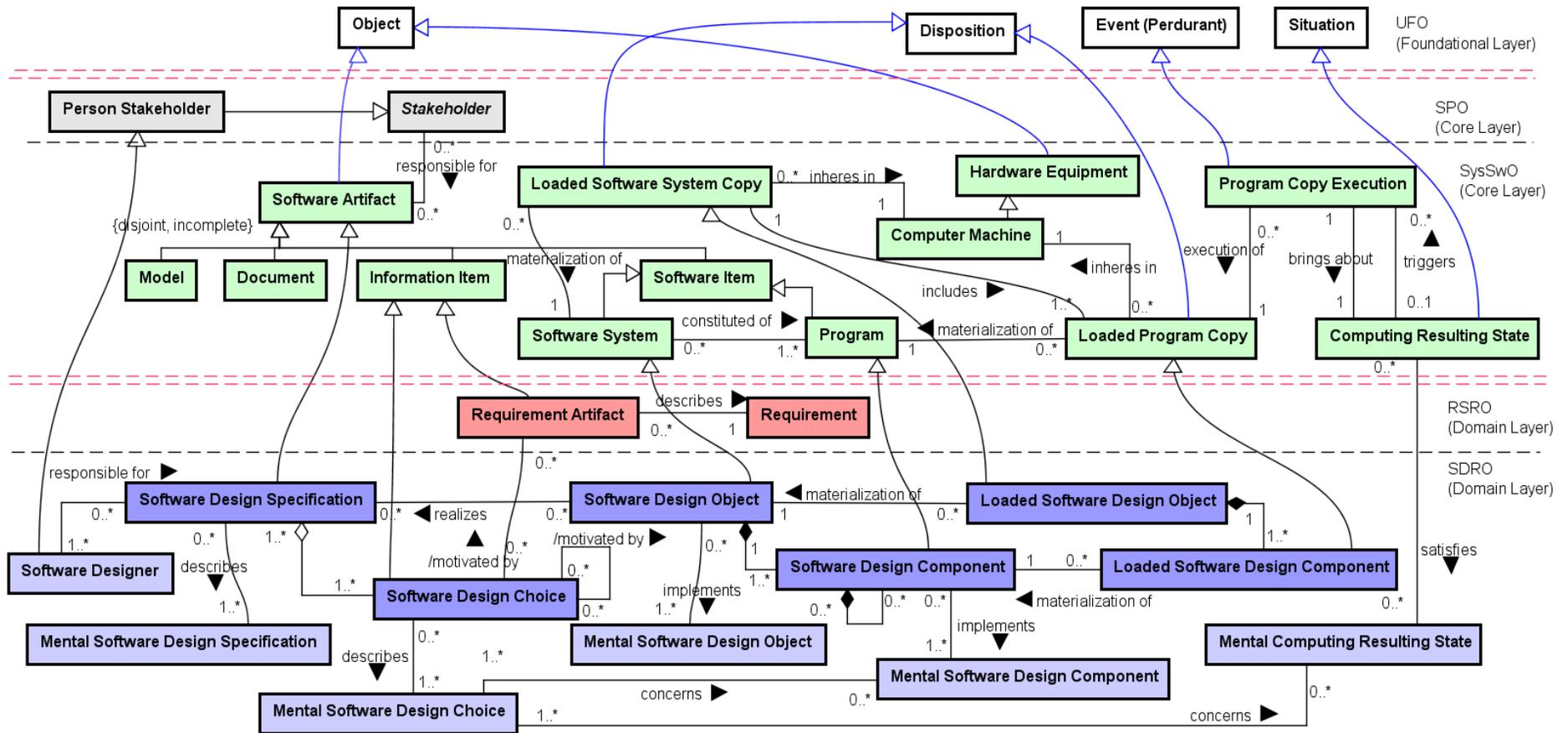


Figure 5.4 – SDRO Physical Aspects sub-ontology conceptual model.

Mental aspects are treated in SDRO as *Software Designer Mental Moments* that inhere in a *Software Designer*, which is a **Person Stakeholder** that uses his/her skills to directly contribute to the outcome of the design effort. The propositional content of a *Software Designer Mental Moment* is a *Software Design Proposition*, i.e., a sentence describing an idea in the designer's mind about software design aspects of a certain object. Thus, *Software Designer Mental Moments* represent mental properties of a *Software Designer* that enable him/her to imagine possible solutions for software design problems, while *Software Design Propositions* are the expression of such solutions in a conversation or in the internal dialogue of the designer. The realization of design solutions is given by situations in reality where the propositional content of *Software Designer Mental Moments* (i.e., *Software Design Propositions*) are true. The number of design solutions that can be actually realized varies according to the intersection between desirable situations that satisfy the requirements and feasible situations that respect the constraints (e.g., time, cost and computational resources) (BAKER; HOEK, 2006).

Five different types of *Software Design Propositions* were identified, based on what their subject refers to. A *Mental Software Design Object* concerns the software object being designed, which may not exist yet (i.e., it represents what designers visualize as the final design object when they refer to “the system”). A *Mental Software Design Object* is specified by a *Mental Software Design Specification*, a detailed description of the object structure (i.e., how the object should be decomposed and organized in smaller elements and how these elements interact with each other). In the beginning of the design process, this description usually represents the object structure in a higher level of abstraction (e.g., architecture aspects) and as design iterations occur, it is refined into more detailed representations (e.g., components and modules) at lower levels of abstraction (PRESSMAN; MAXIM, 2020). The *Mental Software Design Specification* consists of one or more *Mental Software Design Choices* made by the *Software Designer*. Each *Mental Software Design Choice* can be motivated by **Requirements** or by other *Mental Software Design Choices* and contains details about a decision made by the *Software Designer* concerning structural or behavioral properties of the designed object or about its components and their connections. Thus, a *Mental Software Design Choice* may concern *Mental Software Design Components*, which represent what the *Software Designer* expects to exist as a part of the designed object in a particular place, playing a specific role and having its own properties (e.g., modules, partitions and layers in which the system's architecture is organized), also referred as “conventional system components” by Guarino

and Melone (2015). A *Mental Software Design Choice* may also concern a *Mental Computing Resulting State*, which represents an expected result of a *Mental Software Design Choice* that can only be assessed in runtime (e.g., obtaining a certain return code after the execution of a system's module).

SDRO physical aspects, in turn, are treated as sub-types of **Software Artifact**. For example, a *Software Design Object* is a **Software System** that implements one or more *Mental Software Design Objects*. This does not imply that every *Mental Software Design Object* results in the development of a new Software System. An existing **Software System** is also considered a *Software Design Object* when it implements at least one *Mental Software Design Object* specified by a *Mental Software Design Specification*. *Software Design Objects* are composed of *Software Design Components*, which are **Programs** that play specific roles in the designed **Software System**, implementing *Mental Software Design Components*. *Software Design Components* can be composed of sub-components, allowing the representation of more complex architectures. A (sub)component can also be part of more than one component (e.g., in situations where the code is properly modularized and reused).

In order to be executed and used, a *Software Design Object* and its *Software Design Components* must be, respectively, materialized as a *Loaded Software Design Object* (Loaded Software System Copy) and *Loaded Software Design Components* (Loaded Program Copies) inhering in a **Computer Machine**, i.e., the software must be loaded in the computer's main memory. A *Loaded Software Design Component* can be executed as a **Program Copy Execution** that brings about a **Computing Resulting State**. If the program was implemented and executed correctly, this **Computing Resulting State** may satisfy a *Mental Computing Resulting State* associated in a *Mental Software Design Choice* with the corresponding *Mental Software Design Component* materialized by the executed *Loaded Software Design Component*. This relationship allows us to verify if the implemented software meets the design specification.

Another type of **Software Artifact** is a *Software Design Specification*, which is created by one or more *Software Designers* and can be either a **Model** (e.g., a class diagram), a **Document** (e.g., a detailed textual description) or a **Software Item** (e.g., a functional prototype), providing an explicit representation that describes *Mental Software Design Specifications* (also referred as Software Design Descriptions in IEEE 1016 (IEEE, 2009)). A *Software Design Specification* is an aggregation of *Software Design Choices*, which are **Information Items** describing *Mental Software Design Choices*.

Therefore, a *Software Design Choice* is a piece of information that physically represents choices made by a *Software Designer* and can be used for communication and evaluation purposes (e.g., a sentence like “*The system will be implemented in Java*” or details added in a class diagram that indicates how entities and relations should be represented in the database). As a derived relation, a *Software Design Choice* is motivated by (and, thus, intends to satisfy) a **Requirement Artifact** when the *Mental Software Design Choice* it describes is motivated by the **Requirement** described by the **Requirement Artifact**. This connection establishes a traceability relation between design and requirements artifacts.

5.2.1 SDRO Evaluation

To evaluate SDRO, we performed Ontology Verification & Validation (V&V) activities by using two approaches: assessment by human approach and data-driven approach (BRANK; GROBELNIK; MLADENIĆ, 2005). In the first, we performed a verification activity by means of expert judgment, in which we checked if the concepts and relations defined in SDRO are able to answer the competency questions. In the second, to validate SDRO, we instantiated its concepts and relations using data extracted from a real-world scenario. Table 5.2 presents the results of SDRO verification, which showed that the ontology answers all the CQs and, thus, is able to cover the scope established to it.

Table 5.2 – SDRO verification against its CQs.

CQs	Description, Concepts and <i>Relations</i>
CQ1	How does a software designer reason about the object being designed? Software Design Proposition is the <i>propositional content</i> of a Software Designer Mental Moment that <i>inherits in</i> a Software Designer , a role played by a Person Stakeholder . Mental Software Design Object , Mental Software Design Specification , Mental Software Design Choice , Mental Software Design Component and Mental Computing Resulting State are <i>subtypes of Software Design Proposition</i> .
CQ2	What is a software design specification? Software Design Specification is a Software Artifact <i>created by Software Designers</i> that <i>describes Mental Software Design Specifications</i> .
CQ3	Which are the components of a software design specification? Software Design Specification <i>is composed of Software Design Choices</i> , which are Information Items that <i>describe Mental Software Design Choices</i> .
CQ4	What is a software design object? Software Design Object is a role played by a Software System that <i>implements Mental Software Design Objects</i> .
CQ5	Which are the components of a software design object? Software Design Object <i>is composed of Software Design Components</i> . Software Design Component is a role played by a Program that <i>implements Mental Software Design Components</i> . It can be <i>composed of</i> other Software Design Components .

Table 5.2 (continuation) – SDRO verification against its CQs.

CQs	Description, Concepts and <i>Relations</i>
	What is described in a software design specification?
CQ6	Software Design Specification <i>describes</i> a Mental Software Design Specification which <i>specifies</i> a Mental Software Design Object and <i>is composed of</i> Mental Software Design Choices , which may <i>concern</i> to Mental Software Design Components or Mental Computing Resulting States .
CQ7	What is the motivation for a software design choice? Software Design Choice <i>describes</i> Mental Software Design Choices , which are <i>motivated by</i> Requirements or by other Mental Software Design Choices . As derived relations, Software Design Choice is <i>motivated by</i> Requirement Artifacts or by other Software Design Choices .
CQ8	How can a software design object be implemented from a software design specification? Software Design Specification <i>is composed of</i> Software Design Choices that <i>describe</i> Mental Software Design Choices <i>concerning</i> Mental Software Design Components . Mental Software Design Components are <i>implemented as</i> Software Design Components , which are <i>components of</i> a Software Design Object that <i>realizes</i> the Software Design Specification .
CQ9	How can a software design object be evaluated against a software design specification? Software Design Object is <i>materialized as</i> a Loaded Software Design Object , which is a role played by a Loaded Software System Copy <i>composed of</i> Loaded Software Design Components . Loaded Software Design Component is a role played by a Loaded Program Copy that <i>materializes</i> a Software Design Component and can be <i>executed</i> in a Program Copy Execution . A Program Copy Execution <i>brings about</i> a Computing Resulting State that can <i>satisfy</i> Mental Computing Resulting States <i>concerned by</i> Mental Software Design Choices <i>described by</i> Software Design Choices <i>encoded in</i> a Software Design Specification that is <i>realized by</i> the Software Design Object .

For SDRO validation, we took as an example of design object the car rental software system (here referred to as *CRS*) specified in (FALBO, 2018) and used SDRO to instantiate and analyze a scenario considering the *CRS* design. The scenario is described below, showing that SDRO can represent real-world situations like that.

The *CRS* system aims to support rent-a-car companies in managing fleets of cars and rentals, as well as allowing customers to make car rentals via internet. Based on this context, the following **Requirements** of *CRS* had been elicited: (i) a rent-a-car company wants to manage fleets of cars; (ii) a rent-a-car company wants to manage customers; (iii) a rent-a-car company wants to manage car rentals; and (iv) customers want to make car rentals via internet. These **Requirements** were described in a document as **Requirement Artifacts**, which is reproduced in Figure 5.5:

Functional Requirements			
Id	Description	Priority	Depends on
FR01	The system should be able to register cars.	High	
FR02	The system should be able to register customers.	High	
FR03	The system should allow customers to make car rentals.	High	FR01, FR02, NFR01

Non-functional Requirements					
Id	Description	Category	Scope	Priority	Depends on
NFR01	The car rental and customer registration functionalities should be available to be accessed via internet from major browsers.	Portability	System	High	

Figure 5.5 – Documented requirements of the CRS system.

Two fictional software engineers, John and Mary, were responsible for designing *CRS* (i.e., playing the role of *Software Designers*) and discussed how they would address these requirements in the system’s implementation. In their discussions, they referred to the software system being designed (*Mental Software Design Object*) as “the system” or “*CRS*”, corresponding to what they had in mind (*Software Designer Mental Moments*) as a solution to satisfy the **Requirements**. In order to treat the *NFR01* requirement, John made some *Mental Software Design Choices* and communicated them to Mary, proposing to implement the system in *Java* using libraries that can run in different browsers (*Google Chrome, Mozilla Firefox* and *Internet Explorer*) and to implement the user interface (a *Mental Software Design Component* in this context) as a separate component independent from the rest of the application. These *Mental Software Design Choices* were also related to *Mental Computing Resulting States* corresponding to John’s visualization of the system’s user interface being executed in each of those different browsers. Mary agreed with his suggestions and complemented that they could organize the system in an architecture based on a combination of partitions and layers: “Fleet Control Partition” would address *FR01*, and “Customer Service Partition” would address *FR02* and *FR03*, corresponding to the part of the system that should be available via internet. Both partitions were composed of three layers: “Presentation Layer”, addressing the interaction between the system and users; “Business Layer”, containing the functionality that support business processes; and “Data Layer”, managing the data access for the application. Partitions and layers were *Mental Software Design Components* which were related to each other in Mary’s *Mental Software Design Choices*. She considered that the combination of the *Mental Software*

Design Choices made by her with the ones made by John (i.e., Mary's *Mental Software Design Specification*) was sufficient to start implementing the system. However, John was having trouble in understanding the choices proposed by Mary (i.e., his *Mental Software Design Specification* was not equivalent to hers), so he asked her to produce a *Software Design Specification* describing what she had in mind. Mary presented him a diagram (shown in Figure 5.6) encoding *Software Design Choices* (e.g., the representation of the partitions and layers as UML elements) describing the *Mental Software Design Choices* she had in mind.

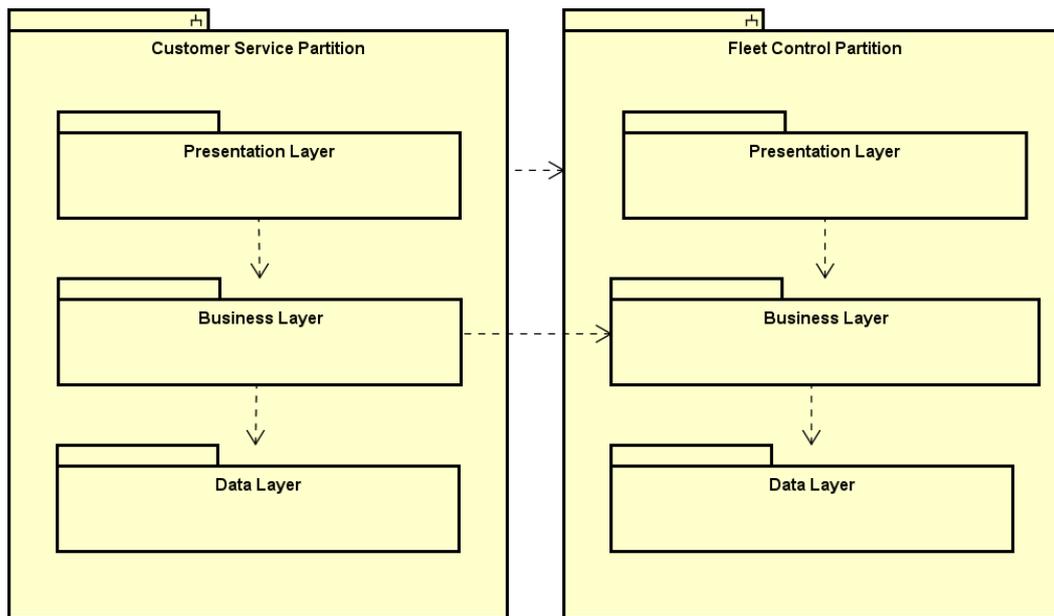


Figure 5.6 – Software architecture proposed by Mary to the *CRS* system.

After seeing the diagram in the *Software Design Specification* produced by Mary, John understood what she proposed (i.e., their *Mental Software Design Specifications* became equivalent) and then they decided to implement the system. They presented the *Software Design Specification* to a developer and explained what they had in mind to him. The interpretation of the developer produced a *Mental Software Design Specification* in his mind, as well as other *Software Design Propositions* associated with it. After that, the developer produced a *Software System* written in *Java*, which satisfied the specification he had in mind (i.e., a *Software Design Object* implementing the *Mental Software Design Object* specified by his *Mental Software Design Specification*). Then, he asked John and Mary to assess if the implemented software corresponded to what they had designed. John and Mary first inspected the code and observed that the partitions and layers had been correctly implemented as *Software Design Components* (i.e., the *Software Design Object* realized the *Software Design Specification*). In the sequence, they loaded a copy

of the system in the computer’s main memory (a *Loaded Software Design Object* composed of *Loaded Software Design Components*) and accessed the system’s user interface through all three browsers previously specified. Each of those accesses produced a **Program Copy Execution** of the *Loaded Software Design Component* that materialized the implementation of the “Presentation Layer” of the “Customer Service Partition” (i.e., a *Mental Software Design Component*), which, in turn, might had triggered the execution of the other components. When the execution was completed, it brought about a **Computing Resulting State** (e.g., a HTTP response code 200) that satisfied the *Mental Computing Resulting State* imagined by John. Based on that, John and Mary concluded that the *Software Design Object* had been correctly implemented according to their *Mental Software Design Objects*.

Table 5.3 presents a summary with some instances of SDRO concepts extracted from the *CRS* example.

Table 5.3 – SDRO instantiation for the CRS system.

Concept	Instance
Software Designer	John; Mary
Requirement (REQ)	REQ1. “A rent-a-car company wants to manage fleets of cars.” REQ2. “A rent-a-car company wants to manage customers.” REQ3. “A rent-a-car company wants to manage car rentals.” REQ4. “Customers want to make car rentals via internet”
Requirement Artifact (RA)	RA1. The description of R1 expressed by <i>FR01</i> . RA2. The description of R2 expressed by <i>FR02</i> . RA3. The description of R3 expressed by <i>FR03</i> . RA4. The description of R4 expressed by <i>NFR01</i> .
Software Designer Mental Moment	Mental properties in the minds of John and Mary expressed by Software Design Propositions.
Software Design Proposition / Mental Software Design Object	“The <i>CRS</i> system”
Software Design Proposition / Mental Software Design Specification	MSDCH1 + MSDCH2
Software Design Proposition / Mental Software Design Choice (MSDCH)	MSDCH1. “to implement the system in <i>Java</i> using libraries that can be run in different browsers.” MSDCH2. “to organize the system in an architecture based on a combination of partitions and layers”
Software Design Proposition / Mental Software Design Component (MSDC)	MSDC1. “Customer Service Partition”. MSDC1.1. “Presentation Layer” from the “Customer Service Partition”. MSDC1.2. “Business Layer” from the “Customer Service Partition”. MSDC1.3. “Data Layer” from the “Customer Service Partition”. MSDC2. “Fleet Control Partition”. MSDC2.1. “Presentation Layer” from the “Customer Service Partition”. MSDC2.2. “Business Layer” from the “Customer Service Partition”. MSDC2.3. “Data Layer” from the “Customer Service Partition”.

Table 5.3 (continuation) – SDRO instantiation for the CRS system.

Concept	Instance
Software Design Proposition / Mental Computing Resulting State (MCRS)	MCRS1. “The system’s user interface accessible from the <i>Google Chrome</i> browser.” MCRS2. “The system’s user interface accessible from the <i>Mozilla Firefox</i> browser.” MCRS3. The system’s user interface accessible from the <i>Internet Explorer</i> browser.
Document / Software Design Specification	A document containing SDCH1 + SDCH2
Software System / Software Design Object	The implementation in <i>Java</i> of the <i>CRS</i> system, consisting of SDC1 + SDC1.1 + SDC1.2 + SDC 1.3 + SDC2 + SDC2.1 + SDC 2.2 + SDC 2.3.
Program / Software Design Component (SDC)	Programs written in <i>Java</i> implementing the mental software design components, including: SDC1. The implementation of MSDC1. SDC1.1. The implementation of MSDC1.1. SDC1.2. The implementation of MSDC1.2. SDC1.3. The implementation of MSDC1.3. SDC2. The implementation of MSDC2. SDC2.1. The implementation of MSDC2.1. SDC2.2. The implementation of MSDC2.2. SDC2.3. The implementation of MSDC2.3.
Loaded Software System Copy / Loaded Software Design Object	A copy of the <i>CRS</i> system loaded in the main memory of the computer used by John and Mary to assess the implementation, consisting of LSDC1 + LSDC1.1 + LSDC1.2 + LSDC 1.3 + LSDC2 + LSDC2.1 + LSDC 2.2 + LSDC 2.3.
Loaded Program Copy / Loaded Software Design Component (LSDC)	Copies of the software design components loaded in the main memory of the computer used by John and Mary to assess the implementation, including: LSDC1. The materialization of SDC1. LSDC1.1. The materialization of SDC1.1. LSDC1.2. The materialization of SDC1.2. LSDC1.3. The materialization of SDC1.3. LSDC2. The materialization of SDC2. LSDC2.1. The materialization of SDC2.1. LSDC2.2. The materialization of SDC2.2. LSDC2.3. The materialization of SDC2.3.
Program Copy Execution (PCE)	PCE1. The execution of LSDC1.1 triggered when the system is accessed through the <i>Google Chrome</i> browser in the computer used by John and Mary to assess the implementation. PCE2. The execution of LSDC1.1 triggered when the system is accessed through the <i>Mozilla Firefox</i> browser in the computer used by John and Mary to assess the implementation. PCE3. The execution of LSDC1.1 triggered when the system is accessed through the <i>Internet Explorer</i> browser in the computer used by John and Mary to assess the implementation.
Computing Resulting State (CRS)	CRS1. The <i>HTTP</i> response code 200 brought about by PCE1. CRS2. The <i>HTTP</i> response code 200 brought about by PCE2. CRS3. The <i>HTTP</i> response code 200 brought about by PCE3.

5.2.2 Discussion

Since SDRO is a networked ontology of SEON, it can also be used as a conceptual framework to discuss design aspects in a wider software development context, exploring questions such as: when a design succeeds or fails, the role of design documentation and the relation between software design and human-computer interaction.

A design effort is considered complete in SDRO when the design specification is realized, which can be satisfied by the following condition: if there are Software Design Components implementing all Mental Software Design Components concerned in all Mental Software Design Choices that are described in Software Design Choices of a Software Design Specification, then we can say that this Software Design Specification is realized by the Software Design Object composed of those Software Design Components. Hence, an incomplete design occurs when at least one Mental Software Design Component is not implemented. However, a complete design does not mean necessarily that the software is correctly implemented, since Software Design Choices that prescribe behaviors of the system can only be assessed when the system is running. Therefore, the correct implementation occurs when all Mental Computing Resulting States concerned by these choices are satisfied by at least a Computing Resulting State. An incomplete or incorrect implementation may happen when a programmer does not follow what was described in the Software Design Specification. In this case, the programmer interpretation of the Software Design Choices probably was not the same as the Designer's. Another reason could be that the Software Design Specification does not describe properly the Mental Software Design Specification created by the Designer, maybe because the tools and the language used to create the specification were not adequate (BAKER; HOEK, 2006).

In SDRO, it is possible to represent a Software System (Design Object) developed without the existence of any physical Software Design Specification. This is the case where Ralph and Wand (2009) describe that the specification is presented as the Design Object itself. Although doing so could be considered a bad practice in software development, it addresses simpler situations where the designer creates the specification only in his mind and develops the system by himself or communicates the design verbally to the developer. Since software development often involves teams and more complex systems, the use of artifacts to represent design specifications is essential to evaluation in early stages and communication of design ideas between designers and other stakeholders. Moreover, it also provides a form of reflexive conversation where the designer can have insights of improvements as he looks to the specification (SCHÖN, 1983; SIMON, 1996).

Finally, it is important to highlight what makes software and software design unique compared to other fields involving design: there is a large gap between what is produced as the Software Design Object and what is perceived by the user in his/her interaction experience. The Software Design Object, a Software System constituted by code (DUARTE *et al.*, 2018), does not interact directly with the user as does a car or a house, for example. It must be loaded in a computer system (i.e., Loaded Software Design Object and Loaded Software Design Components) and then executed, so the user can interact with the result of this execution (i.e., Computing Resulting States). Therefore, software design methods, tools and languages should consider not only the internal structural aspects of software but also its external characteristics exhibited to the user. Traditionally, Software Engineering research is more focused on the former, while the latter is relegated to Human-Computer Interaction (HCI) studies (TAYLOR; VAN DER HOEK, 2007). Thus, to reduce the gap between software design and user experience, it is essential to have a holistic view of design and look at software as a whole. The Human-Computer Interaction Design Ontology (HCIDO) aims to help in reducing this gap, merging design aspects from Software Engineering and HCI areas. In next section, we present HCIDO.

5.3 Human-Computer Interaction Design Ontology

HCI design is focused on how to design an interactive computer system to support users to achieve their goals through the interaction between them and the system (SUTCLIFFE, 2014). The Human-Computer Interaction Design Ontology (HCIDO) is proposed in this work to provide a well-founded consensual conceptualization about the HCI design of interactive systems. In its current version, HCIDO is focused on the design of interactive software systems, rather than interactive computer systems (i.e., it does not address the design of hardware aspects). HCIDO addresses the knowledge intersection between SE and HCI domains by connecting software design concepts from SDRO (a domain ontology of SEON) with HCI core concepts from HCIO (a core ontology of HCI-ON). Just like SDRO, HCIDO scope was defined by means of competency questions. The set of the identified competency questions (CQ) is presented in Table 5.4. HCIDO specializes SDRO to the HCI design domain. Thus, as it can be noticed in Table 5.4, the competency questions defined to HCIDO are, in fact, SDRO competency questions specialized to the HCI context.

Table 5.4 – HCIDO Competency Questions

Competency Question	Rationale
CQ1. How does an HCI designer reason about the object being designed?	Understand the mental nature of HCI design.
CQ2. What is an HCI design specification? CQ3. Which are the components of an HCI design specification?	Understand what an HCI design specification is and its constituents.
CQ4. What is an HCI design object? CQ5. Which are the components of an HCI design object?	Represent the system built from an HCI design specification and its components.
CQ6. What is described in an HCI design specification?	Understand the design-related information described in an HCI design specification and the relation between the HCI design specification of an object and the mental elements of the design of that object.
CQ7. What is the motivation for an HCI design choice?	Understand the reason why a designer makes a design choice regarding an HCI design object.
CQ8. How can an HCI design object be implemented from an HCI design specification?	Understand how the information encoded in an HCI design specification is transformed into an implemented HCI design object.
CQ9. How can an HCI design object be evaluated against an HCI design specification?	Understand how the information encoded in an HCI design specification can be used to evaluate the implementation of an HCI design object.

HCIDO reuses the distinction between mental and physical aspects from SDRO and is focused on characterizing specific aspects related to the design specification and the design object, which have differences in the HCI design context in relation to the software design domain. In order to highlight this difference between HCIDO and SDRO, we decomposed HCIDO into two sub-ontologies: the *Design Specification* sub-ontology and the *Design Object* sub-ontology. The architecture of HCIDO is presented in Figure 5.7.

HCIDO *Design Specification* sub-ontology is presented in Figure 5.8 and *Design Object* sub-ontology is presented in Figure 5.9. In the figures, red double-dashed lines separate the layers of SEON and HCI-ON architectures, according to the classification proposed by Scherp *et al.* (2011). The black single-dashed horizontal lines separate concepts from different ontologies at the same layer. The same colors used in SDRO (Figure 5.3 and Figure 5.4) are used to indicate concepts from different ontologies and the name of each ontology is indicated in the figures next to the right border. HCIO concepts are presented in different colors to indicate from which sub-ontology each concept comes from, according to the modularization presented in Figure 5.7. Concepts from HCIDO are presented in different shades of orange according to the sub-ontology to which they belong: light orange is used for *Design Specification* sub-ontology and dark orange for *Design Object* sub-ontology. In the model description, SEON concepts are written in **bold**, SDRO concepts are in ***bold italics***, while HCI-ON concepts are underlined and HCIDO concepts are *underlined in italics*.

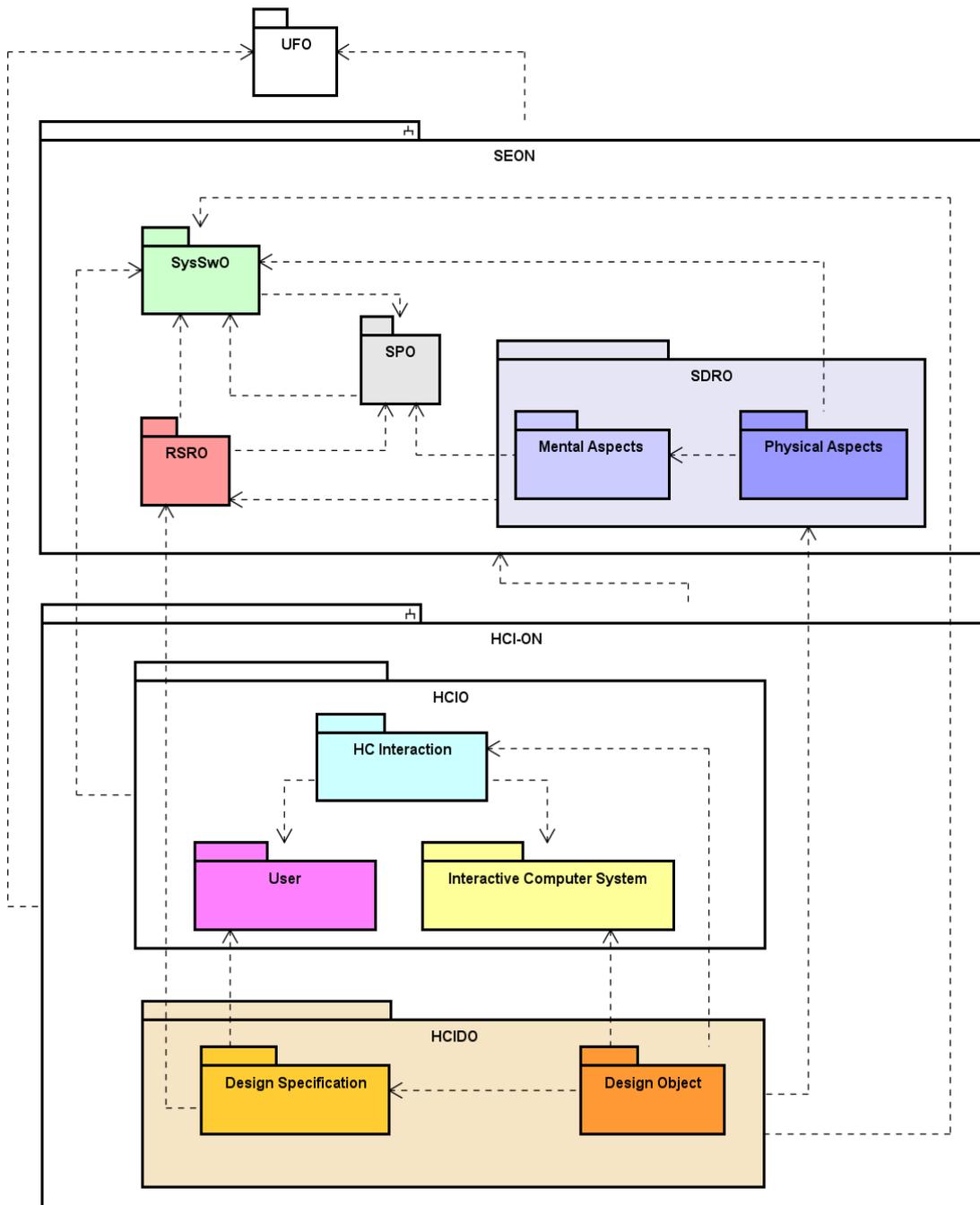


Figure 5.7 – HCIDO architecture.

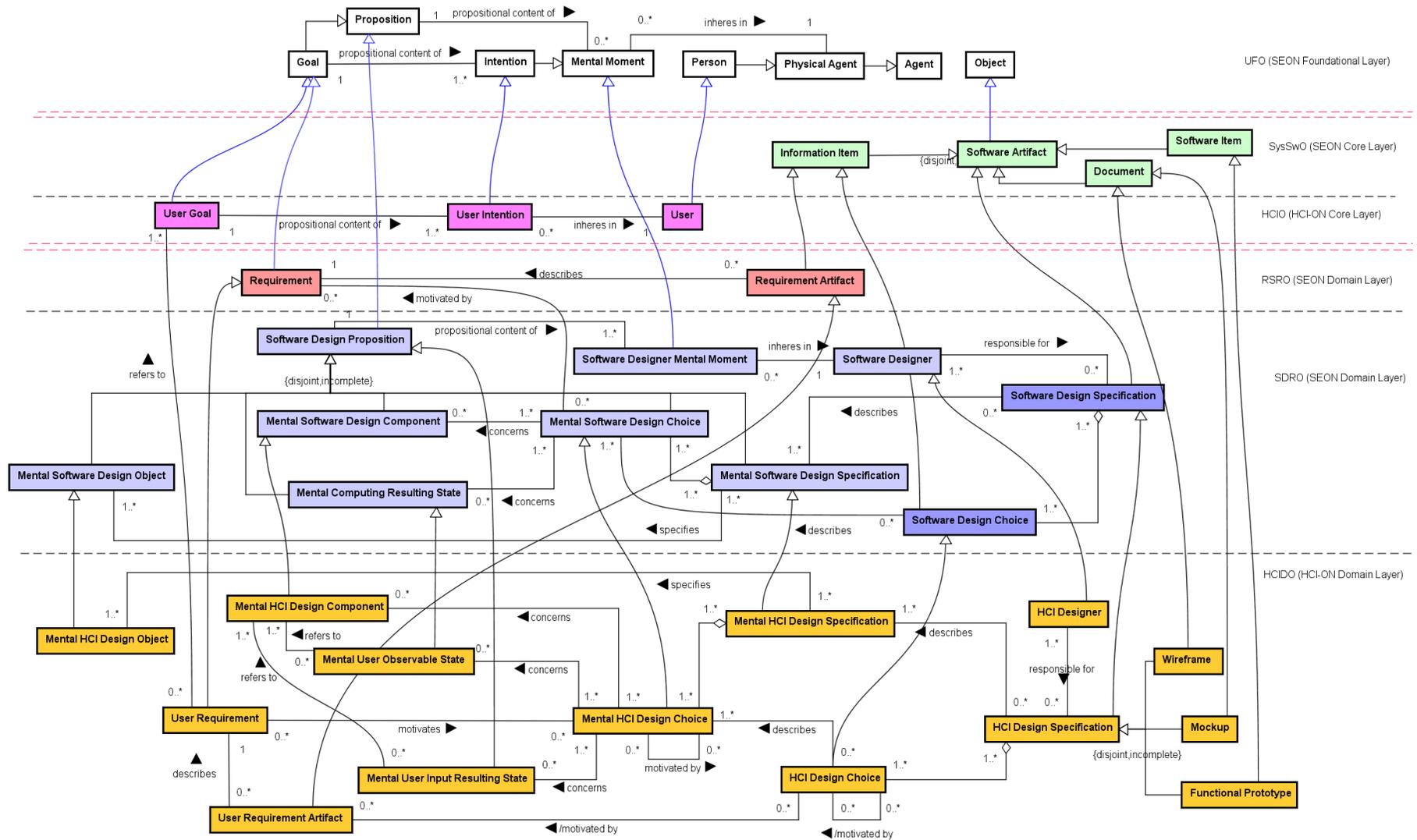


Figure 5.8 – HCIDO Design Specification sub-ontology conceptual model.

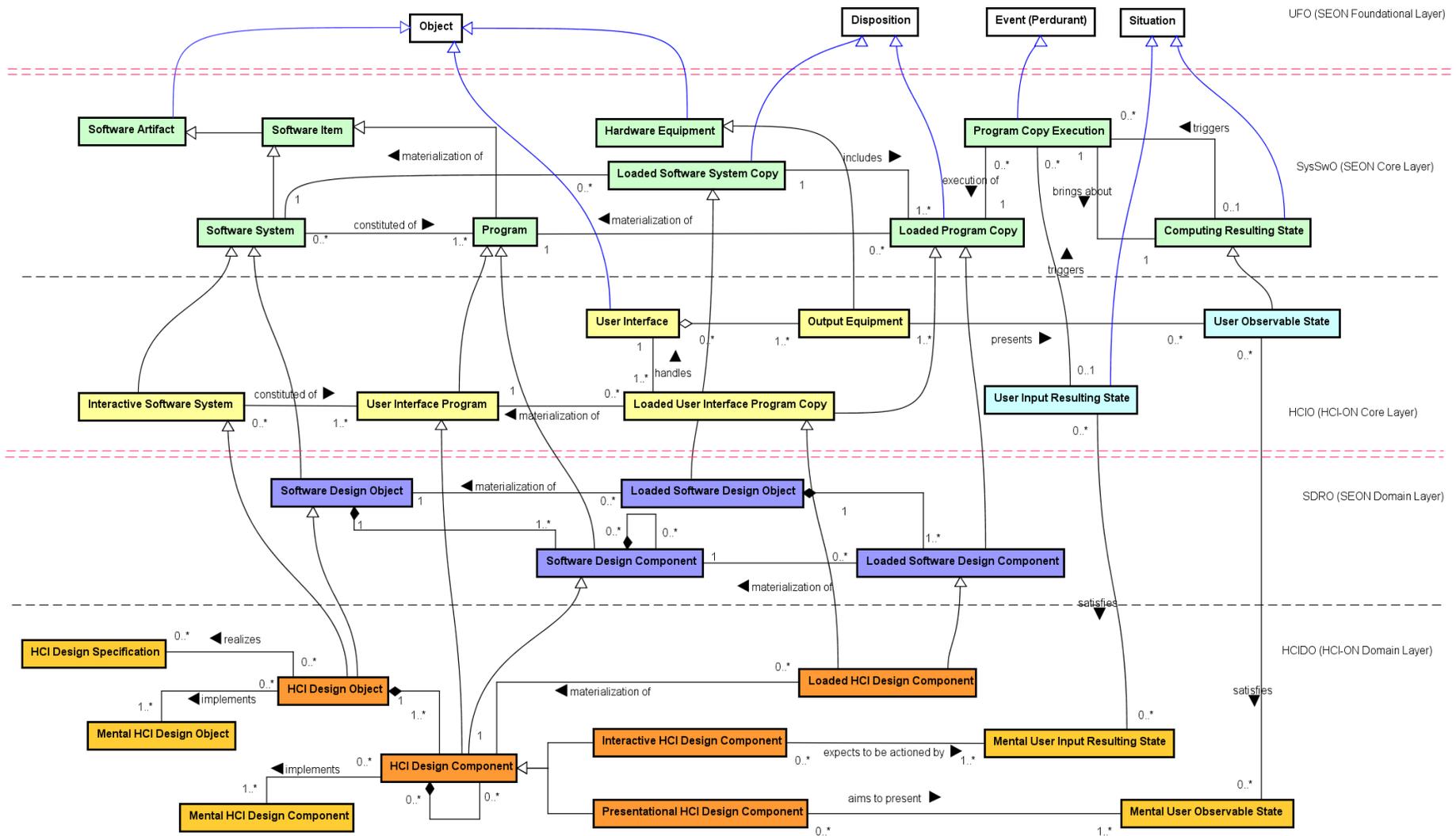


Figure 5.9 – HCIDO Design Object sub-ontology conceptual model.

An *HCI Designer* is a **Software Designer** that uses his or her skills to directly contribute to the creation of an *HCI Design Specification*, a particular type of **Software Design Specification** that deals with human-computer interaction aspects. The mental aspects of HCIDO are treated as subtypes of **Software Design Propositions** which are the propositional content of **Software Designer Mental Moments** inhering in *HCI Designers*. A *Mental HCI Design Choice* is a **Mental Software Design Choice** defining how the human-computer interaction should be implemented, including aspects related to the system's appearance, the disposition of components in space and time and their expected behaviors in response to user actions.

Mental HCI Design Choices can be motivated by previous *Mental HCI Design Choices* or by *User Requirements*, which are **Requirements** that refer to *User Goals* (i.e., **Requirements** concerned with users' needs or capabilities that should be addressed by the system in order to allow users to achieve their goals in an effective, efficient, safe and satisfying manner). *User Requirement Artifacts* are **Requirement Artifacts** that describe *User Requirements* (e.g., user requirements written as user stories). When performing an HCI design process based on ISO 9241-210 (ISO, 2019), for example, *User Requirement Artifacts* are usually produced in the activity "specifying the user requirements". It is important to highlight that the motivation for the *Mental HCI Design Choices* is not always explicit in real-world situations (e.g., when design choices are motivated by designer's tacit knowledge).

The content of a *Mental HCI Design Choice* may be a general choice regarding the system (e.g., the definition of which colors should be used in the system's interface), or a specific choice related to *Mental HCI Design Components*, which are **Mental Software Design Components** that can be perceived or actioned by users (e.g., a text label and a button) through the user interface. A *Mental HCI Design Choice* related to a *Mental HCI Design Component* that can be perceived by users is also associated to *Mental User Observable States*, which are **Mental Computing Resulting States** describing how one or more *Mental HCI Design Components* should be presented to users. For example, in a *Mental HCI Design Choice* of "displaying products in a paginated list with 12 products per page", the product and the list are *Mental HCI Design Components*. The product may have a default layout and an alternative one that is used when some field is empty (e.g., presenting a product without picture in a different manner). These two layouts are described by *Mental User Observable States* resulting from *Mental HCI Design Choices* associated to the same *Mental HCI Design Component* (i.e., the product). Conversely, a *Mental HCI Design Choice* related to a *Mental HCI Design Component* that can be actioned by users is associated to *Mental User Input Resulting States*, which are

Software Design Propositions describing situations, conditions or constraints related to actions that users can or cannot perform in some Mental HCI Design Components (e.g., a button that can only be clicked after filling all required fields in a form, a text input field that accepts only numbers).

A Mental HCI Design Specification is a **Mental Software Design Specification** consisting of a set of Mental HCI Design Choices. It represents ideas that give form to a detailed description in the designer's mind about HCI structural and behavioral aspects of the interactive system. These ideas can be encoded in one or more HCI Design Specifications, which is a **Software Design Specification** describing Mental HCI Design Specifications (e.g., a hand-drawn sketch or a text document). HCI Design Specifications encode one or more HCI Design Choices, which are **Software Design Choices** that describe Mental HCI Design Choices (e.g., the fragment of a sketch showing the fields of a form arranged in two columns or a sentence written in a document describing the expected behavior after a form submission). Hence, HCI Design Choices are the physical representation of Mental HCI Design Choices, which can be used for communication and evaluation purposes.

Three subtypes of HCI Design Specifications are defined in HCIDO: Wireframes, Mockups and Functional Prototypes. A Wireframe is a **Document** outlining the basic structure of the interactive system's user interface (e.g., how elements are visually organized when displayed at the screen) in a low fidelity sketch, which does not address specific details such as colors and typography. A Mockup, in turn, is a higher fidelity **Document** depicting how the interactive system should be presented to users, similar to screenshots of the system's future screens. Finally, a Functional Prototype is a piece of code (i.e., a **Software Item**) intended to present basic functionality of an interactive system or of its components. It is developed for early evaluation purposes and cannot be considered the final implementation. We decided to represent only these three types because our focus was to illustrate some kinds of HCI design specifications, and we did not intend to create a complete taxonomy. Moreover, the survey results presented in Chapter 4 showed that these were the most popular artifacts produced by HCI designers. In a design process, it is common that low fidelity artifacts are used in initial steps and are refined into higher fidelity artifacts as feedback is provided by other stakeholders and the solution gets more mature.

The design object is addressed as an Interactive Software System in HCIDO. Although HCI design involves both the design of software and hardware elements of an interactive computer system, HCIDO addresses HCI design in the context of software development, thus hardware aspects are treated as (non-functional) **Requirements**. Hence,

a Mental HCI Design Object is a **Mental Software Design Object** specified by one or more Mental HCI Design Specifications. HCI Design Specifications that describe Mental HCI Design Specifications are realized by one or more HCI Design Objects, which are Interactive Software Systems playing the role of **Software Design Objects**, implementing one or more Mental HCI Design Objects. An HCI Design Object is composed of HCI Design Components, which are User Interface Programs that play the role of **Software Design Components** and implement Mental HCI Design Components. Each HCI Design Component has its own structure, appearance and behavior and usually is composed of other HCI Design Components (e.g., a piece of code that implements the user interface of a “product” component, which can be used both in a “list of products” component and in a “shopping cart” component).

HCI Design Components can be classified into two types considering the role they play in the human-computer interaction. A Presentation HCI Design Component (e.g., a text label) implements Mental User Observable States and aims to present information that can be perceived through users’ senses. An Interactive HCI Design Component (e.g., a button), in turn, implements Mental User Input Resulting States and is expected to be actioned or not in certain conditions after actions performed by users of the interactive system. It is important to notice that these two types are not disjoint, i.e., an HCI Design Component can be both Presentational and Interactive. HCI Design Components are materialized as Loaded HCI Design Components, which are Loaded User Interface Program Copies that play the role of **Loaded Software Design Components** (i.e., copies of programs that deal with user interface aspects loaded in the memory of an interactive computer system).

The **Program Copy Execution** of a Loaded HCI Design Component that materializes a Presentation HCI Design Component brings about User Observable States. If a User Observable State satisfies the Mental User Observable State which the component implements, this means that the Presentation HCI Design Component was correctly implemented. Conversely, a User Input Resulting State may trigger a **Program Copy Execution** of a Loaded HCI Design Component that materializes an Interactive HCI Design Component. The implementation of an Interactive HCI Design Component is correct when a User Input Resulting State satisfies a Mental User Input Resulting State implemented by an Interactive HCI Design Component and the triggered **Program Copy Execution** comes from a Loaded HCI Design Component that materializes that same Interactive HCI Design Component.

5.3.1 HCIDO Evaluation

Like SDRO, the evaluation of HCIDO was also performed through Ontology Verification & Validation (V&V) activities using assessment by human and data-driven approaches (BRANK; GROBELNIK; MLADENIĆ, 2005). Verification and validation activities were performed considering, respectively, the evaluation criteria C1 (the ontology elements must be the ones sufficient and necessary to cover the scope defined by means of competency questions) and C2 (the ontology must be able to represent real-world situations), defined during the *Relevance Cycle* of the research method followed in this work, as explained in Chapter 1 (Section 1.4).

Table 5.5 presents the results of verification of HCIDO by means of expert judgment, which showed that the ontology answers all of the CQs and, thus, covers the established scope.

Table 5.5 – HCIDO verification against its CQs.

CQs	Description, Concepts and <i>Relations</i>
CQ1	How does an HCI designer reason about the object being designed? HCI Designer is a Software Designer . Mental HCI Design Object, Mental HCI Design Specification, Mental HCI Design Choice, Mental HCI Design Component, Mental User Observable State and Mental User Input Resulting State are <i>propositional contents of Software Designer Mental Moments</i> that <i>inhere in</i> an HCI Designer .
CQ2	What is an HCI design specification? HCI Design Specification is a Software Design Specification <i>created by HCI Designer</i> that <i>describes</i> Mental HCI Design Specifications . Wireframe and Mockup are Documents and Functional Prototype is a Software Item . They are <i>subtypes of HCI Design Specification</i> .
CQ3	Which are the components of an HCI design specification? HCI Design Specification <i>is composed of</i> HCI Design Choices , which are Software Design Choices that <i>describe</i> Mental HCI Design Choices .
CQ4	What is an HCI design object? HCI Design Object is an Interactive Software System that plays the role of Software Design Object and <i>implements</i> a Mental HCI Design Object .
CQ5	Which are the components of an HCI design object? HCI Design Object <i>is composed of</i> HCI Design Components . HCI Design Component is a User Interface Program that plays the role of Software Design Component and <i>implements</i> a Mental HCI Design Component . It can be <i>composed of</i> other HCI Design Components . Interactive HCI Design Component and Presentational HCI Design Component are <i>subtypes of</i> HCI Design Component . An Interactive HCI Design Component <i>expects to be actioned</i> by Mental User Input Resulting States . A Presentational HCI Design Component <i>aims to present</i> Mental User Observable States .
CQ6	What is described in an HCI design specification? HCI Design Specification <i>describes</i> Mental HCI Design Specification which <i>specifies</i> a Mental HCI Design Object and <i>is composed of</i> Mental HCI Design Choices , which may <i>concern</i> to Mental HCI Design Components, Mental User Observable States and Mental User Input Resulting States .

Table 5.5 (continuation) – HCIDO verification against its CQs.

CQs	Description, Concepts and <i>Relations</i>
CQ7	<p>What is the motivation for an HCI design choice? User Requirement is a Requirement that <i>refers to</i> a User Goal. User Requirement Artifact is a Requirement Artifact that <i>describes</i> a User Requirement. HCI Design Choice <i>describes</i> Mental HCI Design Choices, which are <i>motivated by</i> User Requirements or by other Mental HCI Design Choices. As derived relations, HCI Design Choice is <i>motivated by</i> User Requirement Artifacts or by other HCI Design Choices.</p>
CQ8	<p>How can an HCI design object be implemented from an HCI design specification? HCI Design Specification is <i>composed of</i> HCI Design Choices that <i>describe</i> Mental HCI Design Choices concerning Mental HCI Design Components. Mental HCI Design Components are <i>implemented as</i> HCI Design Components, which are <i>components of</i> an HCI Design Object that <i>realizes</i> the HCI Design Specification.</p>
CQ9	<p>How can an HCI design object be evaluated against an HCI design specification? HCI Design Object is <i>composed of</i> HCI Design Components, which are <i>materialized as</i> Loaded HCI Design Components. Loaded HCI Design Component is a Loaded User Interface Program Copy that plays the role of Loaded Software Design Component. A Loaded HCI Design Component can be <i>executed</i> in a Program Copy Execution, which is <i>triggered by</i> a User Input Resulting State and <i>brings about</i> a User Observable State. User Input Resulting State and User Observable State respectively <i>satisfy</i> Mental User Input Resulting States and Mental User Observable States, which are <i>concerned with</i> Mental HCI Design Choices <i>described by</i> HCI Design Choices <i>encoded in</i> an HCI Design Specification.</p>

For the validation of HCIDO, we took the same car rental software system presented in Section 5.2.1 (the *CRS* system) and used the ontology to instantiate and analyze a scenario considering its HCI design. The scenario is described below, showing that HCIDO can represent real-world situations like that.

After the first version of the *CRS* was implemented, John and Mary invited some potential customers (Users) to perform the task of renting a car using the *CRS* system (User Goal), in order to evaluate the system under the users' point of view. After that, John and Mary asked them to point out improvements that could be made in *CRS*, considering the experience they had with the system. Some of the considerations pointed by the users (User Requirements) were that (i) the car rental form was too extensive and could be broken into a multi-step form and (ii) after filling all the information requested by the form, they were requested to log in or register a new account, and then were redirected to another page, losing the information they had filled in the previous form. Based on that, John and Mary described two additional requirements *NFR02* and *NFR03* (User Requirement Artifacts) in the *CRS* requirements document, which are depicted in Figure 5.10.

Functional Requirements			
Id	Description	Priority	Depends on
FR01	The system should be able to register cars.	High	
FR02	The system should be able to register customers.	High	
FR03	The system should allow customers to make car rentals.	High	FR01, FR02, NFR01

Non-functional Requirements					
Id	Description	Category	Scope	Priority	Depends on
NFR01	The car rental and customer registration functionalities should be available to be accessed via internet from major browsers.	Portability	System	High	
NFR02	The car rental process should be broken into smaller steps, in order to reduce user's cognitive load.	Usability	Functionality	Medium	
NFR03	Customers must be able to log in or register a new account when making a car rental, so they can complete the car rental process more quickly.	Usability	Functionality	High	

Figure 5.10 – User requirements (highlighted in yellow) added to the *CRS* requirements document.

Since John and Mary had limited knowledge about HCI aspects, they asked Mark, an *HCI Designer*, to propose a new version of the system's user interface satisfying those *User Requirements*. Mark had the idea (*Mental HCI Design Choice*) of meeting *NFR02* by grouping related fields (e.g., rental duration, customer info, payment details) from the car rental form and splitting them into different pages, showing a progress indicator at the top of each page (fields, forms, pages and the progress indicator are instances of *Mental HCI Design Component*). In order to meet *NFR03*, he also thought (*Mental HCI Design Choice*) about embedding the sign-up and sign in forms into the car rental form. Then, Mark decided to draw a *Wireframe* (*HCI Design Specification*) sketching these ideas, in order to have a better vision on how these *Mental HCI Design Component* could be visually organized on the screen and also communicate them to John and Mary. The *Wireframe* drawn by Mark is reproduced in Figure 5.11 and is composed of several other pieces of information (*HCI Design Choices*) describing the *Mental HCI Design Choices* he had made, which may have motivated other *Mental HCI Design Choices* displayed in the figure. For example, the choice of inserting the "Have an account?" field was made in the moment he was drawing the *Wireframe* and was motivated by the choice of embedding the sign-up and sign in forms into the car rental form. Some choices did not have a clear and explicit motivation, i.e., they were neither motivated by other previous choices

nor made to meet the specified requirements. That was the case of using a date picker to allow users selecting the pick-up and drop-off dates, which may had been decided based on Mark’s knowledge about best practices, for example.

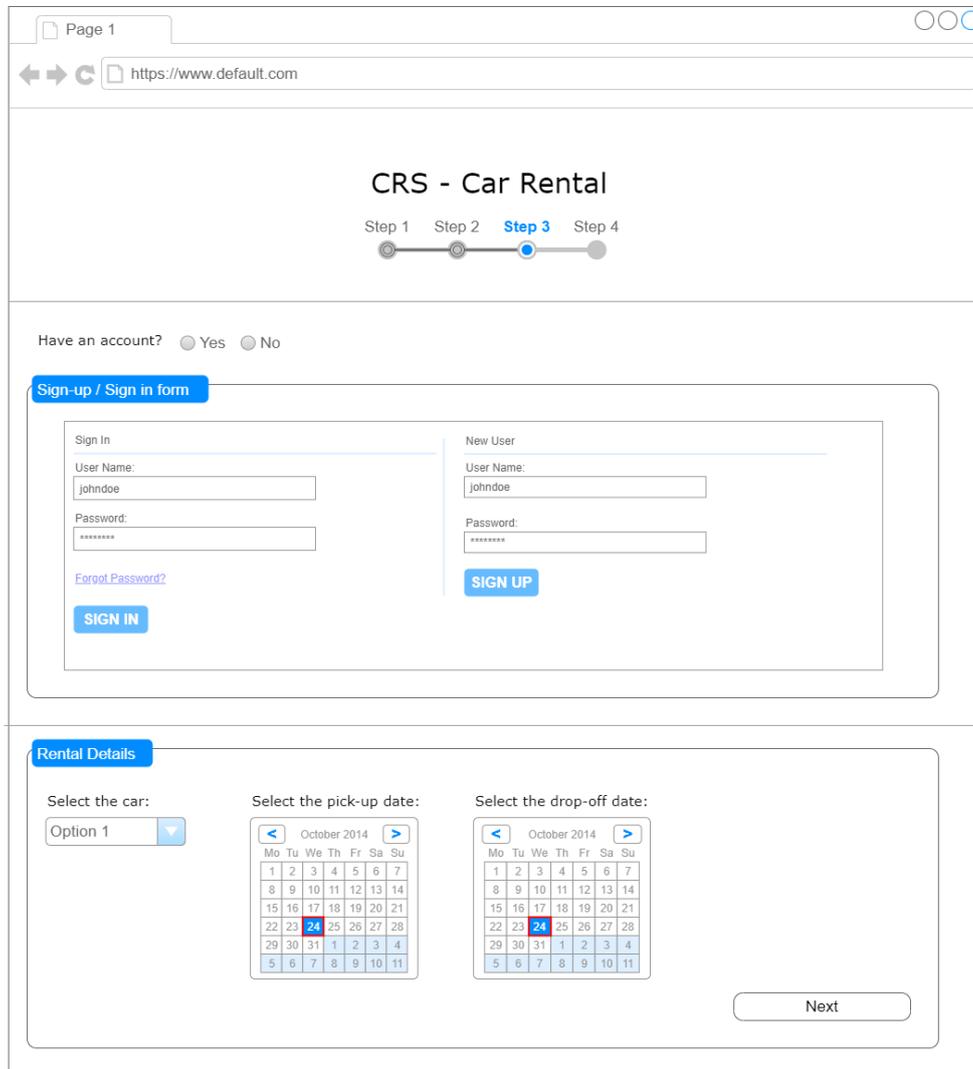


Figure 5.11 – Wireframe proposed by Mark for the *CRS* system.

John and Mary approved Mark’s proposal and asked him to present how the final user interface will look like in a *Mockup*. When Mark was drawing the *Mockup*, he was making *Mental HCI Design Choices* that associated *Mental User Observable States* to *Mental HCI Design Components* (e.g., the submit button would have sharp corners, no borders, blue background and white text). He also decided to use an input mask in the phone field to help users enter their phone numbers in the right format (*Mental User Input Resulting State*). A fragment of the *Mockup* drawn by Mark describing the choices related to the sign-up form is presented in Figure 5.12.

Register account

Please enter the requested data and then press the “Register” button.

Name*

Your name

E-mail*

your@email.com

Phone*

() - - - -

Password*

Register

Figure 5.12 – Fragment of a mockup showing the “Register account” form of the *CRS* system.

Mark also produced a UI kit (presented in Figure 5.13), which was a *Mockup* describing the style of each kind of form component used in the system. For each component, he detailed how it would provide visual feedback about a certain state (*Mental User Observable State*) (e.g., a button can be unclicked, mouse overed, clicked or disabled). The transitions between these states could be actioned by user actions (*Mental User Input Resulting States*), for example the user moving the mouse pointer over the button and the button going from unclicked to mouse overed. A *Functional Prototype* was also developed by Mark, aiming to describe the behavior he imagined for the sign up and sign in forms. He implemented a piece of software in which the sign-up form is displayed when the option “No” for the “Have an account?” field is checked, and the sign in form is displayed when it is set to “Yes”.

After that, John and Mary asked a developer to implement the new user interface based on the artifacts produced by Mark and the developer implemented it using web technologies such as *Java Server Pages*, *HTML*, *CSS* and *JavaScript*. Part of the code produced by the developer for the sign-up form is presented in Figure 5.14 and the implementation of the style of the submit button is detailed in Figure 5.15. Both the implementation of the form and of the button are instances of (*Interactive* and *Presentational*) *HCI Design Components*. The element represented by the “p” *HTML* tag, in turn, is just a *Presentational HCI Design Component*, since it has no interactive behavior.

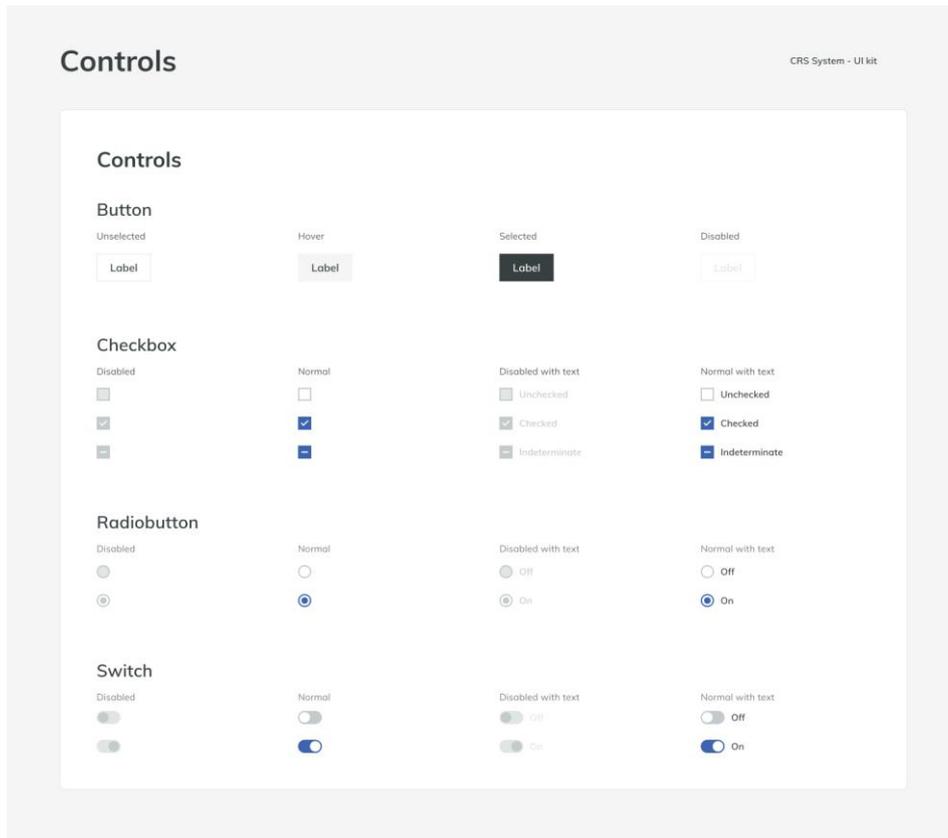


Figure 5.13 – Mockup of the different states of form elements used in the CRS system.

```

...
<form id="signUpForm" action="process.jsp">
  <p>Please enter the requested data and then press the "Register" button.</p>
  <div class="form-control input-name">
    <label for="sign-up__name">Name</label><br>
    <input id="sign-up__name" type="text" name="name" />
  </div>
  <div class="form-control input-email">
    <label for="sign-up__email">E-mail</label><br>
    <input id="sign-up__email" type="text" name="email" />
  </div>
  <div class="form-control input-phone">
    <label for="sign-up__phone">Phone</label><br>
    <input id="sign-up__phone" type="text" name="phone" />
  </div>
  <div class="form-control input-password">
    <label for="sign-up__password">Password</label><br>
    <input id="sign-up__password" type="password" name="password" />
  </div>
  <div class="actions">
    <input id="sign-up__submit" class="btn" type="submit" value="Register" />
  </div>
</form>
...

```

Figure 5.14 – HTML code implementing the user interface of the sign-up form in the CRS system.

```

<style lang="css">
  #sign-up__submit {
    background-color: #162db1;
    color: white;
    border-radius: 0;
    border: none;
  }
</style>

```

Figure 5.15 – CSS code implementing the user interface of the submit button of the sign up form in the CRS system.

In the sequence, the code of the new user interface was integrated into the CRS code base and a new version of the system was deployed, loading the implemented programs (*Loaded HCI Design Components*) in the computer machine of the web server. Then, John and Mary tested it by accessing the system’s user interface in a browser and registering a new account (in this case, they were acting as Users in order to evaluate the system). They observed (User Observable State) that it looked the same way as Mark had depicted in the *Mockup* (i.e., the planned *Mental User Observable States* were satisfied). Moreover, when filling the phone field, John accidentally pressed the “R” key (a User Input Resulting State caused by an unintentional action of the user) and observed that it was not displayed in the input value (User Observable State), thus they perceived that the input mask was working well (i.e., the *Mental User Input Resulting State* designed by Mark was not satisfied by the User Input Resulting State created after his unintentional action).

Table 5.6 presents a summary with some instances of HCIDO concepts extracted from the CRS example.

Table 5.6 – HCIDO instantiation.

Concept	Instance
HCI Designer	Mark
User Requirement (UR)	UR1. “The car rental form was too extensive and could be broken into a multi-step form.” UR2. “After filling all the information requested by the form, users were requested to log in or register a new account, and then were redirected to another page, losing the information they had filled in the previous form.”
User Requirement Artifact (URA)	URA1. The description of UR1 expressed by <i>NFR02</i> . URA2. The description of UR2 expressed by <i>NFR03</i> .
Mental HCI Design Object	“The CRS system”
Mental HCI Design Specification	MHDCH1 + MHDCH2

Table 5.6 (continuation) – HCIDO instantiation.

Concept	Instance
Mental HCI Design Choice (MHDCH)	MHDCH1. “grouping related fields (e.g., rental duration, customer info, payment details) from the car rental form and splitting them into different pages, showing a progress indicator at the top of each page.” MHDCH2. “embedding the sign-up and sign in forms into the car rental form.” MHDCH3. “inserting the ‘Have an account?’ field to control the exhibition of the sign-up and sign in forms.” MHDCH4. “displaying the submit button with sharp corners, no borders, blue background and white text.” MHDCH5. “using an input mask in the phone field to help users entering their phone numbers in the right format.”
Mental HCI Design Component (MHDC)	MHDC1. The car rental form. MHDC2. The sign-up form. MHDC3. The sign in form. MHDC4. The submit button from the sign-up form. MHDC5. The phone field from the sign-up form.
Mental User Observable State	“The submit button with sharp corners, no borders, blue background and white text.”
Mental User Input Resulting State	“A sequence of numbers entered in the phone field that follows the format defined by the input mask.”
HCI Design Specification / Wireframe	The wireframe produced by Mark for the car rental form.
HCI Design Specification / Mockup	The mockup produced by Mark for the sign-up form.
HCI Design Specification / Functional Prototype	The prototype developed by Mark to demonstrate the “Have an account?” field.
HCI Design Choice (HDCH)	HDCH1. The pick-up and drop-off date fields using a date picker displayed in the wireframe. HDCH2. The progress indicator displayed in the wireframe. HDCH3. The forms displayed in the wireframe. HDCH4. The sign-up form displayed in the mockup. HDCH5. The submit button of the sign-up form displayed in the mockup.
HCI Design Object	The implementation of CRS with the new user interface.
HCI Design Component / Interactive HCI Design Component (IHDC)	IHDC1. The form HTML element whose id is “signUpForm”. IHDC2. The input HTML element whose id is “sign-up__submit”.
HCI Design Component / Presentational HCI Design Component (PHDC)	PHDC1. The “p” HTML element in the “signUpForm” containing the instructions to fill the form.
Loaded HCI Design Component	IHDC1, IHDC2 and PHDC1 deployed in the web server and loaded in its computer machine.
User Input Resulting State (UIRS)	UIRS1. The value of the phone field of the sign-up form after John pressing the “R” key.
User Observable State (UOS)	UOS1. The submit button for the sign-up form displayed the same way as it was prescribed by the mockup. UOS2. The phone field of the sign-up form not displaying “R” in its value.

5.3.2 Discussion

The example presented above as an instantiation of HCIDO can be explored to analyze challenges related to communication and knowledge transfer in HCI design. We discuss them below and propose solutions that consider the application of HCIDO.

As the interactive system development process advances, the number of design choices tend to increase more and more. Consequently, if the motivations for design choices are not documented (i.e., transformed into explicit knowledge), in the future there are less chances of remembering the reasons why each choice was made, making it more difficult to reuse that knowledge. Moreover, when the motivation for design choices is not explicit, other stakeholders are hampered to have a complete understanding about what the designer proposed. This happens because some design choices are made based on the designer's tacit knowledge, which cannot be easily articulated. In this scenario, HCIDO can be used to support knowledge representation, integration, search and retrieval. For example, HCIDO can support HCI design teams instantiating information from real HCI design projects, making team members aware of possible mental elements not described in artifacts or elements described in artifacts that do not have a clear motivation. By doing so, the ontology helps link physical information with the mental aspects encoded in people's mind, supporting the transformation of tacit knowledge into explicit knowledge in a knowledge externalization process.

However, even explicit HCI design knowledge encoded in design artifacts (i.e., design specifications) can lead to interpretation conflicts if the purpose of each kind of artifact and the semantics of their elements are not clear, explicit and shared among the team. For instance, in Figure 5.12, the information inside the phone field represents an input mask that should be visible while the user is typing. On the other hand, the information inside the e-mail field represents a placeholder, which aims to give an example of a possible input to users and disappears when the field is focused (i.e., user starts typing). Although they have different semantics, they are syntactically represented in the same way, relying on a shared and implicit understanding between who produced and who consumed the artifact to make the distinction between their meanings. This may occur because some artifacts are not appropriate to describe certain aspects (e.g., wireframes are not so good to represent interactive behaviors), thus the combination of different kinds of artifacts providing different and complementary views about the design solution can be a good strategy. Moreover, HCIDO can be used as a reference framework to provide a better understanding of the meaning of each kind of design specification or to semantically annotate design artifacts to

support knowledge retrieval. Since HCIDO is integrated to HCI-ON and SEON, design artifacts can be annotated using a common structure with documents addressing requirements, code and testing, providing information traceability among these artifacts and supporting knowledge reuse.

5.4 Related Works

The ontologies presented in this chapter are strongly influenced by the works by Ralph and Wand (2009) and Guarino (2014), which discussed ontological aspects of design. Although they provide a good framework to understand the meaning of design in general, they do not explain the ontological distinction between design specifications that are purely mental and design specifications encoded in artifacts, for example. In addition, design in the software development context is different from other fields since it results in an intangible and abstract object (i.e., a program that is coded in a programming language, to be run in computers, resulting in the behaviors prescribed by the program), rather than a physical and tangible object like a car or a house (OSTERWEIL, 2007). Specific aspects related to this abstract nature of the software design object (e.g., how it can be evaluated against a design specification) are also not addressed by these works.

Other important contributions about software design also highlighted some additional aspects that were considered in this work. For example, Baker and van der Hoek (2006) provide a more explicit understanding about the mental and physical elements and their connections, as well as Ralph's Sensemaking-Coevolution-Implementation Theory (RALPH, 2015). Gero's FBS and sFBS (GERO, 1990; GERO; KANNENGIESSER, 2014) ontologies highlight the cognitive and physical processes of transformations between function, behavior and structure that enable designers to create a description of a design solution from a set of requirements. However, they are more general and neither of them go into further details about the design specification and the composition of the design object. Other works, on the other hand, provide ontologies of more specific aspects of software design, namely domain-driven design (SAIYD; SAID; NEAIMI, 2009), model-based design (DE MEDEIROS; SCHWABE; FEIJÓ, 2005) and design intent (SOLANKI, 2015). Hence, they cannot provide a general conceptualization about software design.

Considering ontologies that address the HCI design domain, as we presented in Chapter 2, none of the analyzed works provide a comprehensive conceptualization about HCI design. Nevertheless, we defined the types of HCI design components in HCIDO based on the work by Paulheim and Probst (2013). Taking into account the purposes for using

ontologies in user interface development (PAULHEIM; PROBST, 2010), since HCIDO is a reference ontology, it cannot be used at runtime (unless it is transformed in an operational ontology) and can be used only for improving user interfaces development process. In this sense, Happel *et al.* (2006) proposed a component library based on ontologies aiming to support software reuse, which is similar to the idea behind the computational tool developed in this work and presented in the next chapter. However, their work focuses on the reuse of (implemented) software components, while our work focuses on the representation, dissemination and reuse of knowledge about HCI design choices and HCI design components.

5.5 Concluding Remarks

This chapter presented the HCI Design Ontology (HCIDO), the reference ontology about HCI design proposed in this work, and the Software Design Reference Ontology (SDRO), which was also developed in this work to provide central notions of design in the software context and be reused in HCIDO development. HCIDO provides a well-founded conceptualization of HCI design. It is a domain ontology of HCI-ON and is also integrated to SEON, addressing the knowledge intersection between SE and HCI. Moreover, HCIDO describes mental and physical aspects involved in HCI design, as well as the relationship between them.

In order to demonstrate how HCIDO can be used as a conceptual framework supporting the development of knowledge management solutions for HCI design of interactive systems, we developed a computational tool based on HCIDO. The tool allows HCI designers and other stakeholders to annotate HCI design artifacts with structured information based on HCIDO conceptual model. Hence, when these pieces of information are combined, they provide new explicit knowledge about aspects that were previously only on people's mind. Moreover, by providing structured search mechanisms, the tool allows this knowledge to be retrieved and reused in future HCI design initiatives. The tool is presented in the next chapter.

Chapter 6

KTID: A Computational Tool to Support KM Aspects in HCI Design

This chapter presents the Knowledge Tool for Interaction Design (KTID), a computational tool developed based on HCIDO conceptualization to support KM aspects in HCI Design. Section 6.1 presents the chapter introduction, Section 6.2 presents KTID, and Section 6.3 presents the study carried out to evaluate KTID. Finally, Section 6.4 presents the chapter concluding remarks.

6.1 Introduction

As we presented in chapters 3 and 4, the lack of a common conceptualization about HCI design has been considered as one of the main challenges involved in KM solutions for HCI design. HCIDO, the Human-Computer Interaction Design Ontology presented in Chapter 5, aims to address this challenge by providing a well-founded and consensual conceptualization of HCI design. In this chapter, we present the Knowledge Tool for Interaction Design (KTID), a computational tool to support KM aspects in HCI design, which was developed based on a conceptual model that reuses concepts and relations from HCIDO conceptualization. Since HCI designers in general have not been familiar with ontologies, HCIDO was used to support the conceptual modeling of KTID, abstracting the ontology to the final users of the tool (i.e., HCI designers). We developed KTID in order to meet R7 (the ontology must be used to solve problems), a requirement established in the *Relevance Cycle* of the research method followed in this work, as we presented in Chapter 1 (Section 1.4). The use of HCIDO to develop KTID served as a proof of concept that showed that it is possible to use HCIDO to develop KM solutions. As a proof of concept, the results show that the use of HCIDO is feasible, but they are not enough to indicate that it works in real settings different from the one considered in this dissertation.

Details about the development and evaluation of KTID are presented in the next sections. For readability reasons, we may sometimes refer to “HCI design” as “design” in this chapter, since we are addressing design specifically in HCI context.

6.2 KTID: *Knowledge Tool for Interaction Design*

When analyzing HCIDO conceptual model, we observed that the physical representations of design choices did not reflect the same relationships as their mental counterparts (i.e., the physical representation of Mental HCI Design Choices, Mental HCI Design Components, Mental User Observable States and Mental User Input Resulting States are all collapsed in HCI Design Choices). Moreover, since HCI Design Specifications are aggregations of HCI Design Choices, sometimes it might be hard to easily perceive all choices as parts of the specification in which they are encoded (i.e., HCI Design Specifications are viewed as a whole). This motivated us to develop KTID as a tool to support HCI designers in describing, sharing and retrieving structured information associated to HCI Design Choices made in HCI Design Specifications. By doing so, KTID aims to aid HCI designers in representing, storing, accessing and evaluating knowledge related to HCI design.

The use of HCIDO in the development of KTID contributed to the understanding of the tool application domain (i.e., HCI design) and to the development of KTID conceptual model through the reuse of parts of HCIDO conceptual model. Since our goal was to allow HCI designers to describe tacit knowledge associated with design choices, we considered only concepts and relationships between mental and physical aspects from HCIDO *Design Specification* sub-ontology. Figure 6.1 presents the UML class diagram representing the KTID conceptual model, with the corresponding reused HCIDO concepts. In the figure, classes painted in yellow were derived from HCIDO and classes painted in grey were created to address specific aspects of KTID application context. Green notes highlight HCIDO concepts that were used directly to derive KTID classes, purple notes inform HCIDO and SEON concepts (as respective relationships) that inspired KTID classes. Finally, pink notes describe constraints to assure data integrity.

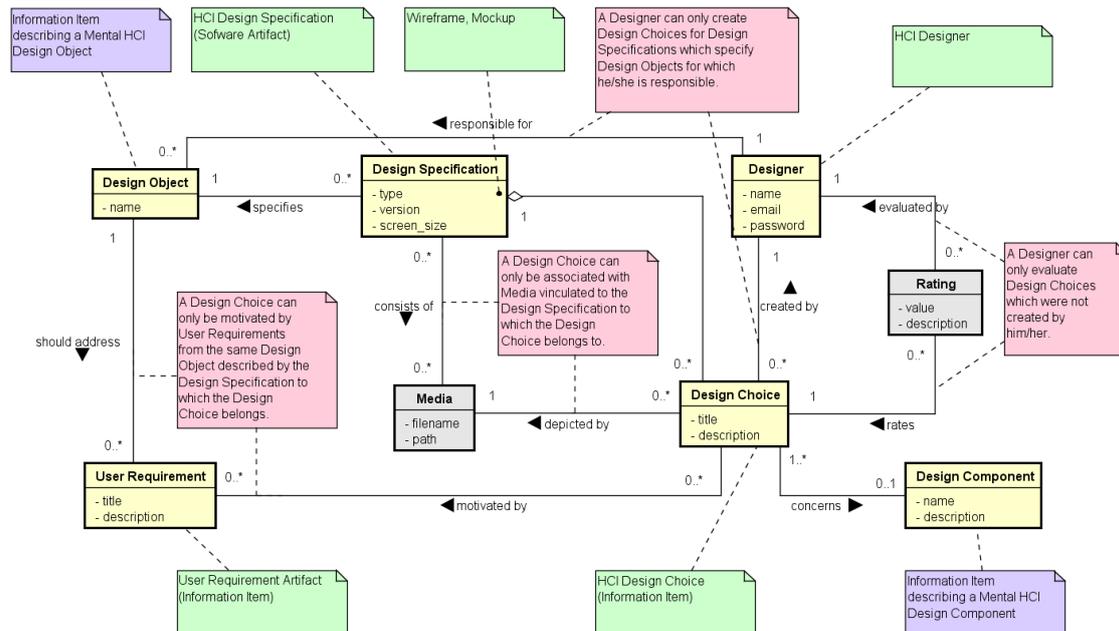


Figure 6.1 – KTID conceptual model and corresponding HCIDO reused concepts.

As for main benefits of using HCIDO in the development of KTID, we point out: (i) we believe that the development of KTID conceptual model based on a general conceptualization of HCI design, rather than on a particular context of the HCI design domain (e.g., HCI design in a specific organization), enabled KTID to be suitable for more HCI design scenarios; and (ii) HCIDO conceptualization allowed us to spend less effort in the conceptual modeling of KTID because we have already obtained knowledge about the domain of interest. HCIDO contributed to our understanding about the HCI design domain, which allowed us to “import” general HCI design concepts and relationships from HCIDO conceptual model, instead of modeling them from scratch. It is worth highlighting that, although we have not used all HCIDO concepts in the KTID conceptual model, HCIDO conceptualization made it clearer for us the existence of mental aspects in HCI design that should be turned into explicit knowledge. As a drawback, we believe that it might be harder to someone less experienced in working with ontologies and ontology networks to understand HCIDO conceptualization, thus he/she might need additional training on this subject.

KTID was developed by a Computer Engineering student in an undergraduate work supervised by the author of this dissertation (OGIONI, 2021). An overview of the main use cases of KTID is presented below through the presentation of some KTID features and screens. HCIDO concepts involved in each use case are written in *italics and underlined*.

Manage Design Objects: this use case aims to record information about *HCI design Objects* being developed and further associate them with their *User Requirements* and *HCI Design*

Specifications. Figure 6.2 presents a screen showing the list of design objects recorded in KTID and Figure 6.3 presents a screen in which we can record a new design object.

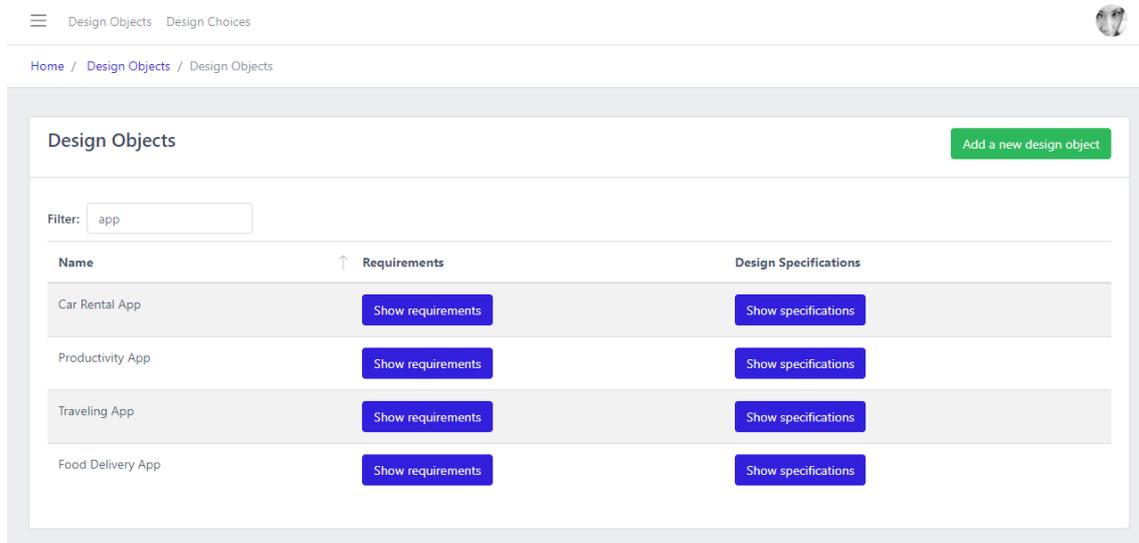


Figure 6.2 – List of design objects recorded in KTID.

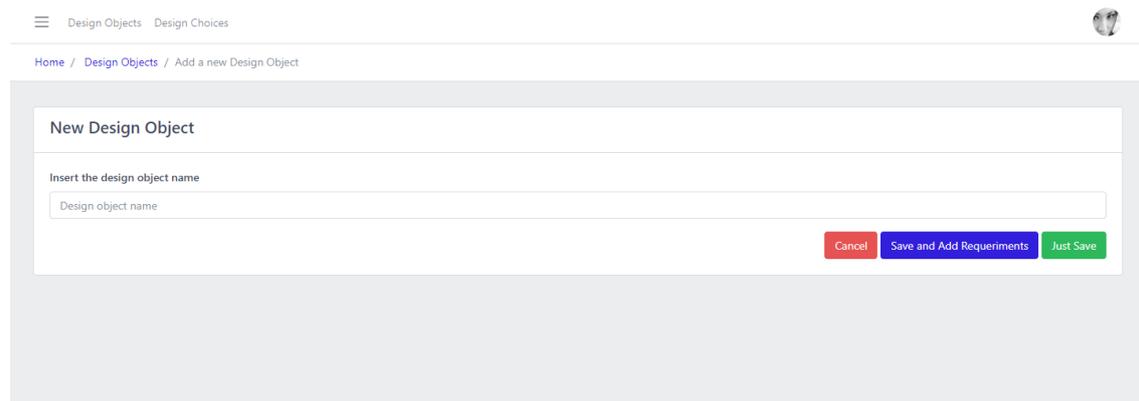


Figure 6.3 – Recording a new design object in KTID.

Manage User Requirements: this use case aims to record information about *User Requirements* associated to an *HCI Design Object* to indicate the user requirements that must be addressed by the HCI design object. Figure 6.4 presents a KTID screen which shows the list of user requirements for a specific design object (in the example, user requirements are represented by means of user stories) and Figure 6.5 presents a screen in which we can record new user requirements.

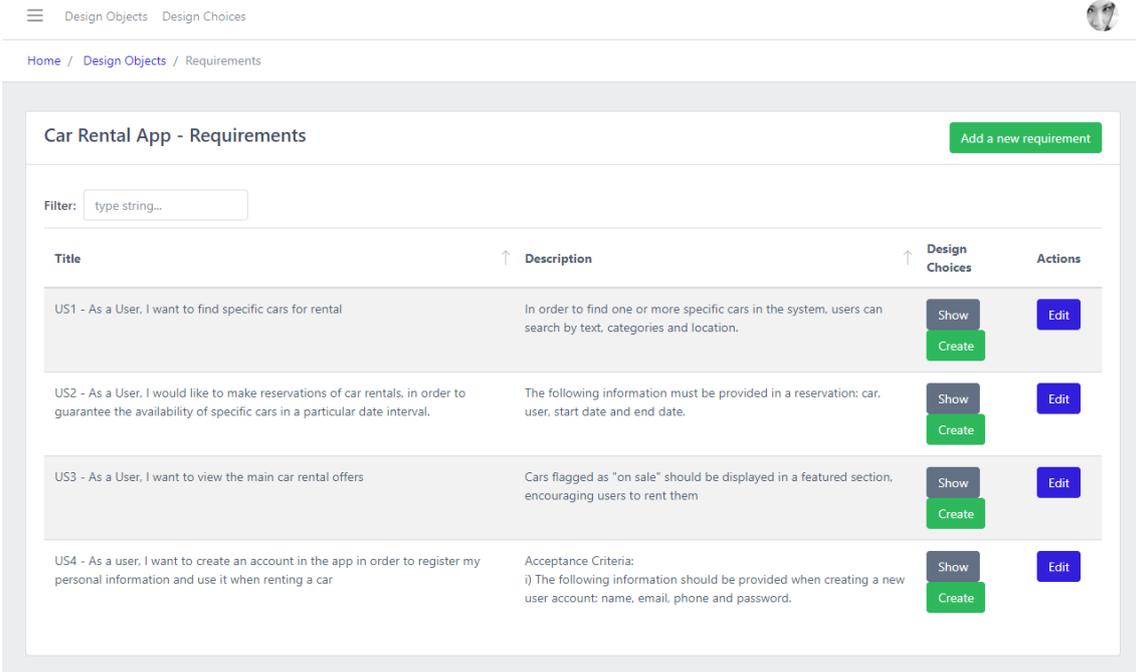


Figure 6.4 – List of user requirements for a design object recorded in KTID.

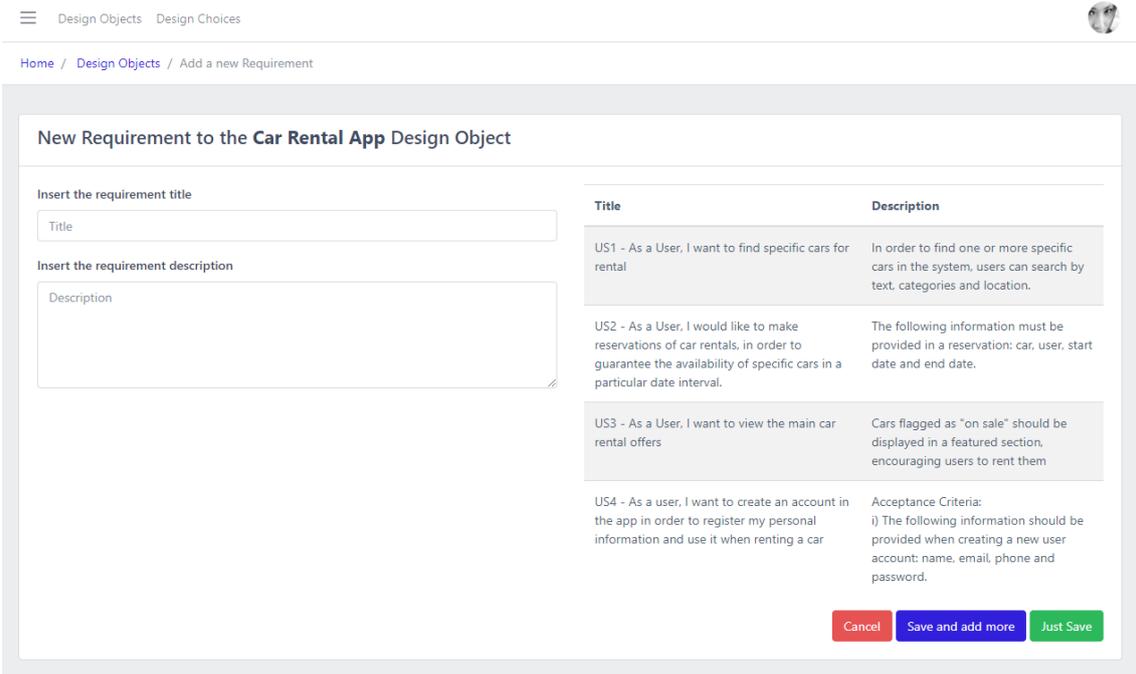


Figure 6.5 – Recording a new user requirement to a design object in KTID.

Manage Design Specifications: this use case aims to record *HCI Design Specifications*, which consist of a set of image files. We can insert several specifications for the same *HCI Design Object*, with different levels of fidelity (e.g., *Mockups*, *Wireframes* or sketches), viewport sizes (e.g., mobile phone, tablets or desktops) and version numbers. Figure 6.6 presents a KTID screen showing the list of files of a design specification as well its design

choices and Figure 6.7 presents the screen in which a new design specification can be recorded.

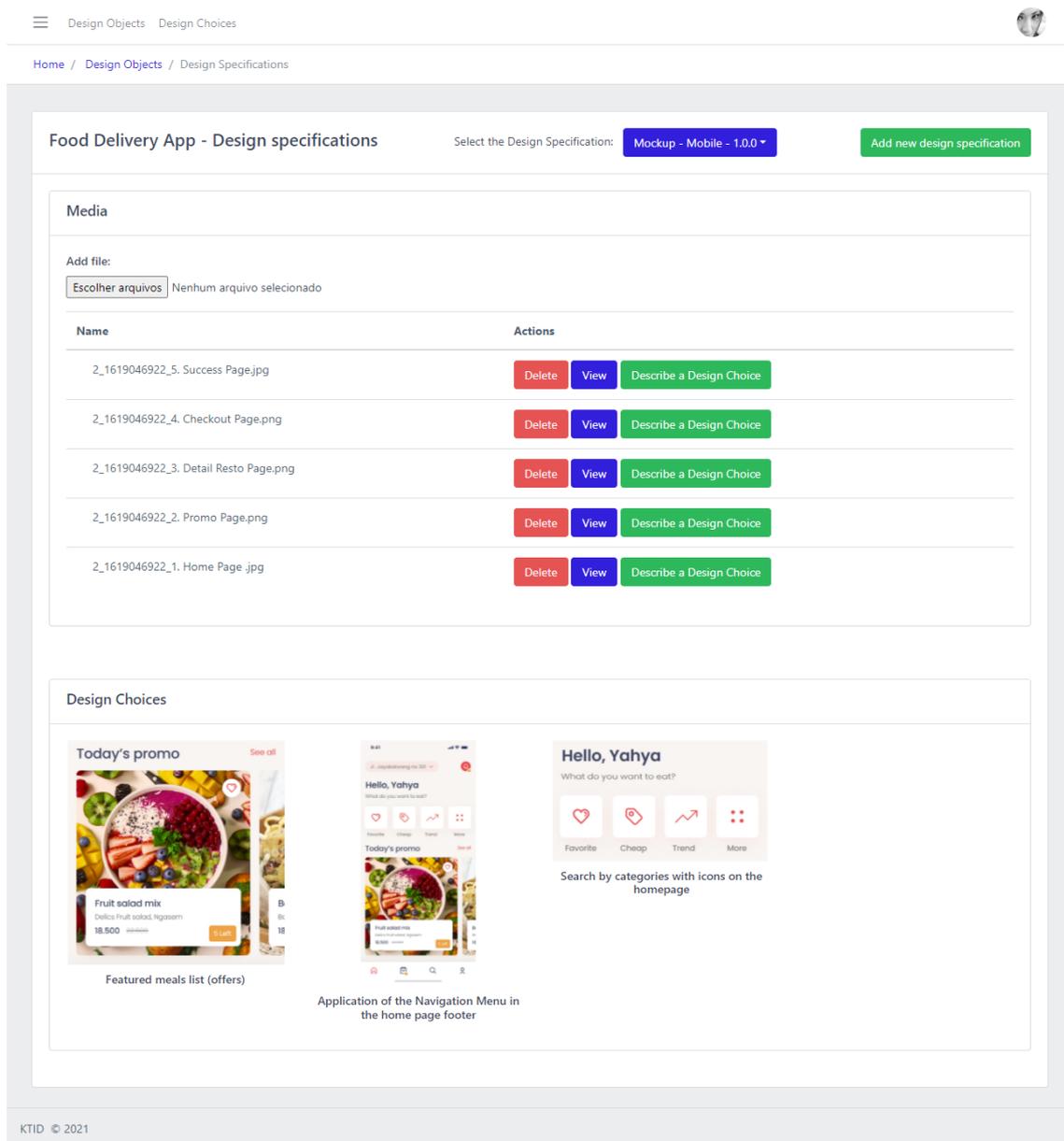


Figure 6.6 – Details of a design specification recorded in KTID.

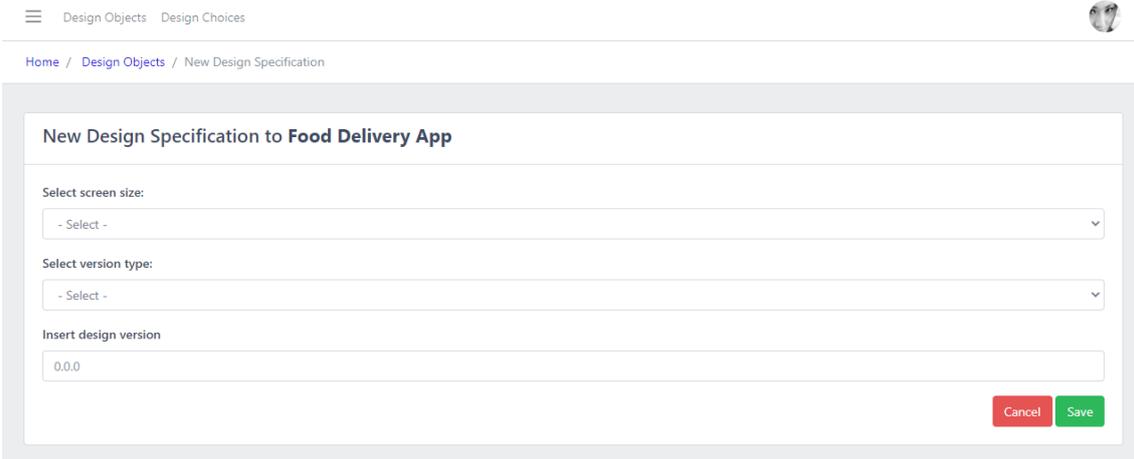


Figure 6.7 – Recording a new design specification for a design object in KTID.

Describe Design Choices: this is one of the main functionalities of KTID. It aims at allowing *HCI Designers* to identify, represent and describe *HCI Design Choices* and related *HCI Design Components* encoded in *HCI Design Specifications*, as well link them with their associated *User Requirements*. Figure 6.8 presents a KTID screen in which a design choice can be described and recorded for a design specification and Figure 6.9 presents a screen showing the details of a design choice recorded in KTID.

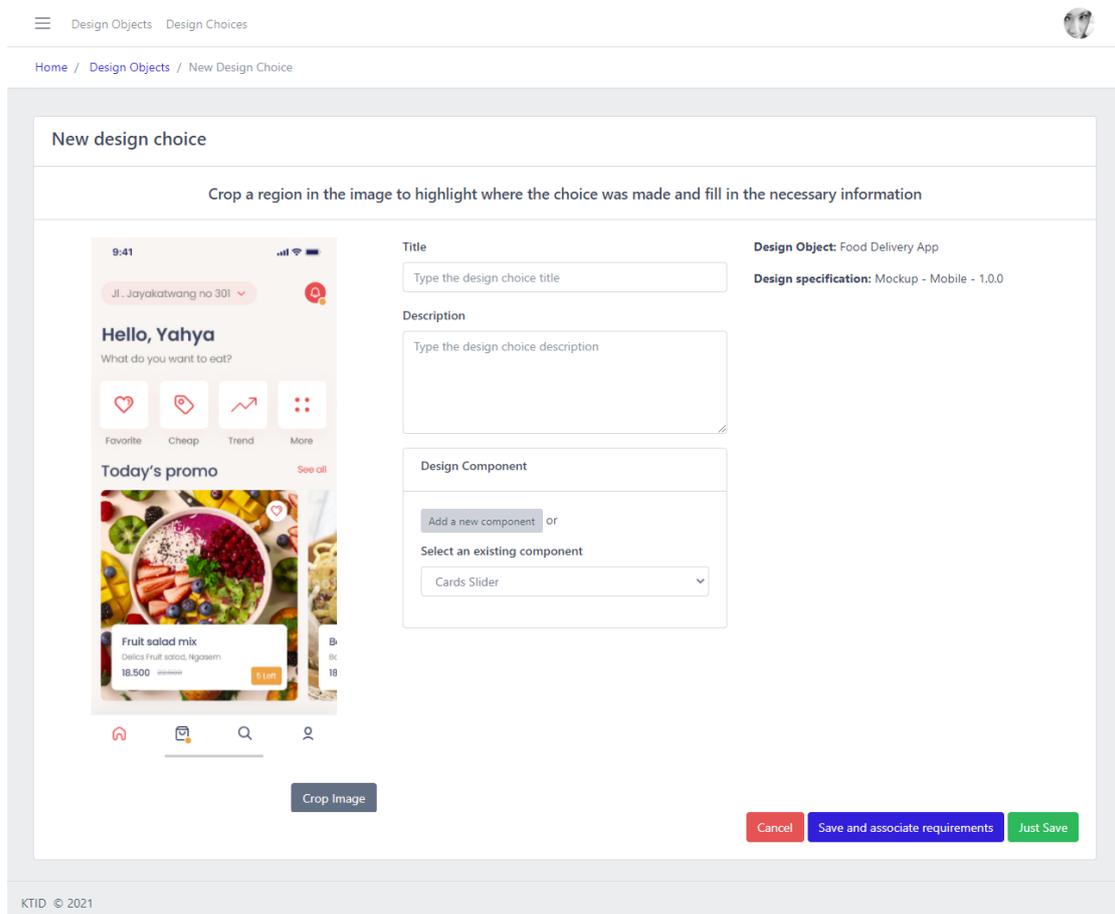


Figure 6.8 – Describing a design choice in KTID.

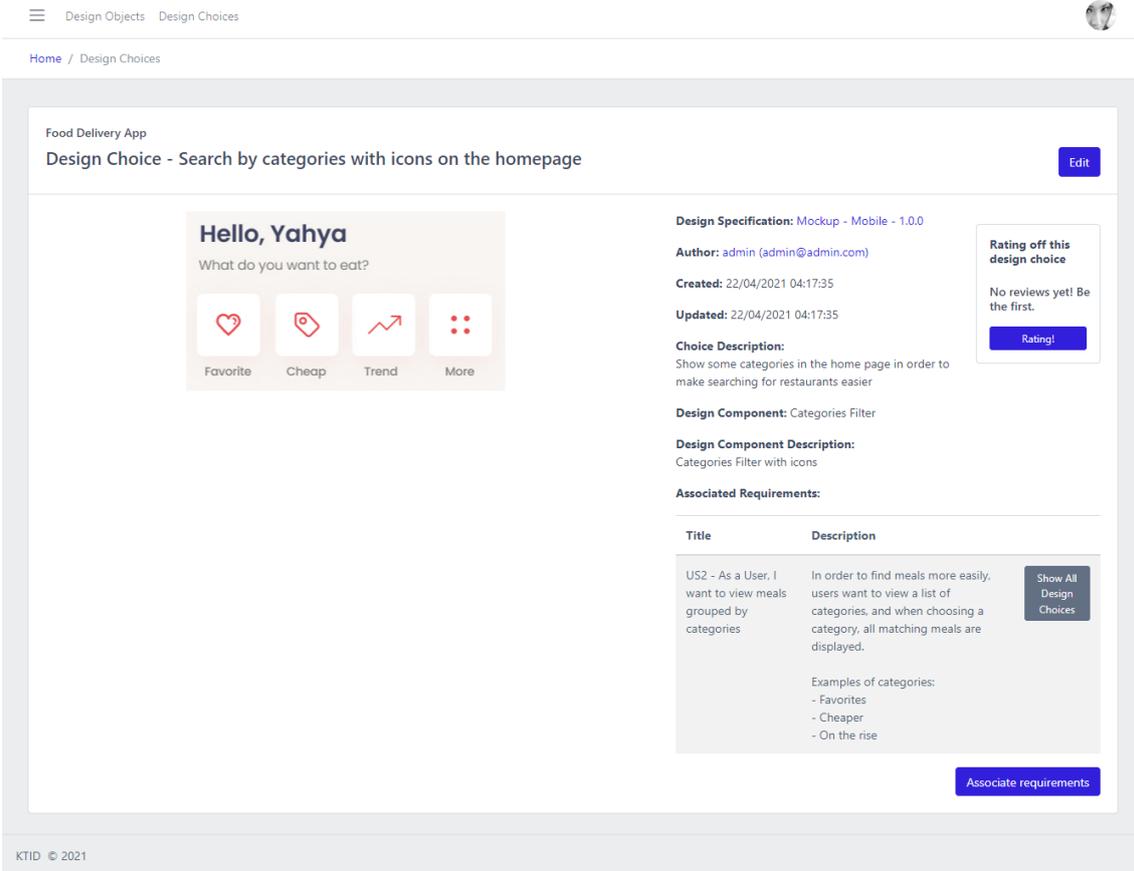


Figure 6.9 – Details of a design choice recorded in KTID.

Search for Design Choices: this is another important functionality of KTID. Figure 6.10 presents a KTID screen in which we can search for all *HCI Design Choices* recorded in KTID. In this screen, we can visualize contextual details (e.g., related components and user requirements, the referred design object and design specification, author, date and time of creation and last update) about each choice. Each column of the table can be filtered or sorted, making it easier to find previous design choices related to contextual information similar to that of another specific context.

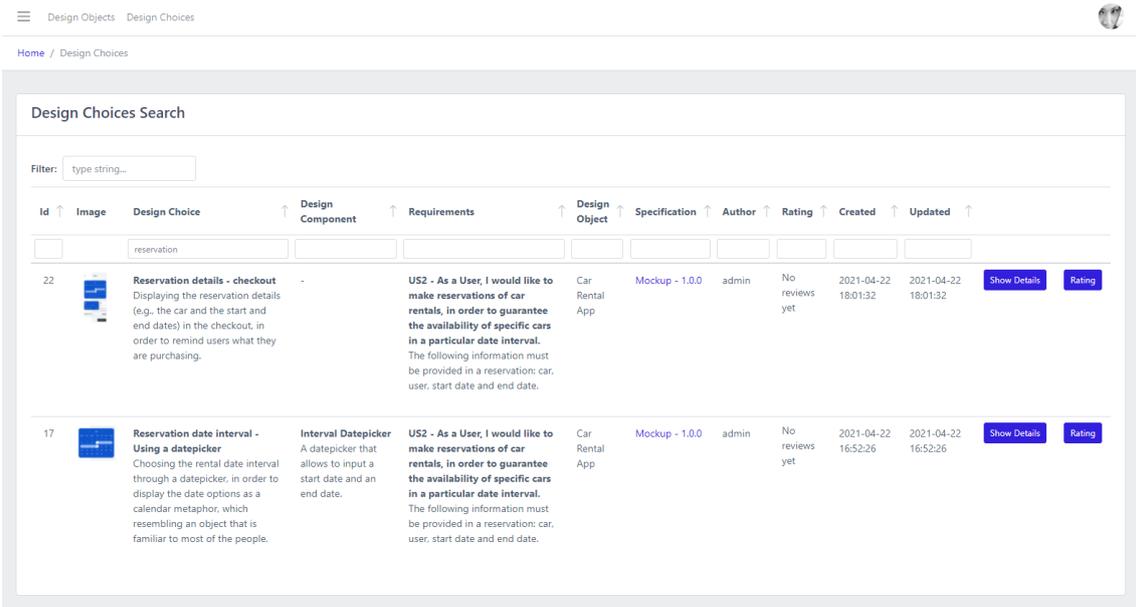


Figure 6.10 – Searching for design choices in KTID.

Evaluate Design Choices: once an *HCI Designer* finds an *HCI Design Choice* satisfying his/her search criteria, he/she can use the design choice and then, evaluate it rating how much it was useful for him/her. By doing so, other designers can make decisions about using or not certain design choice based on its average rating. Figure 6.11 presents a KTID screen in which we can evaluate a design choice and Figure 6.12 presents a screen showing the list of evaluations of a design choice.

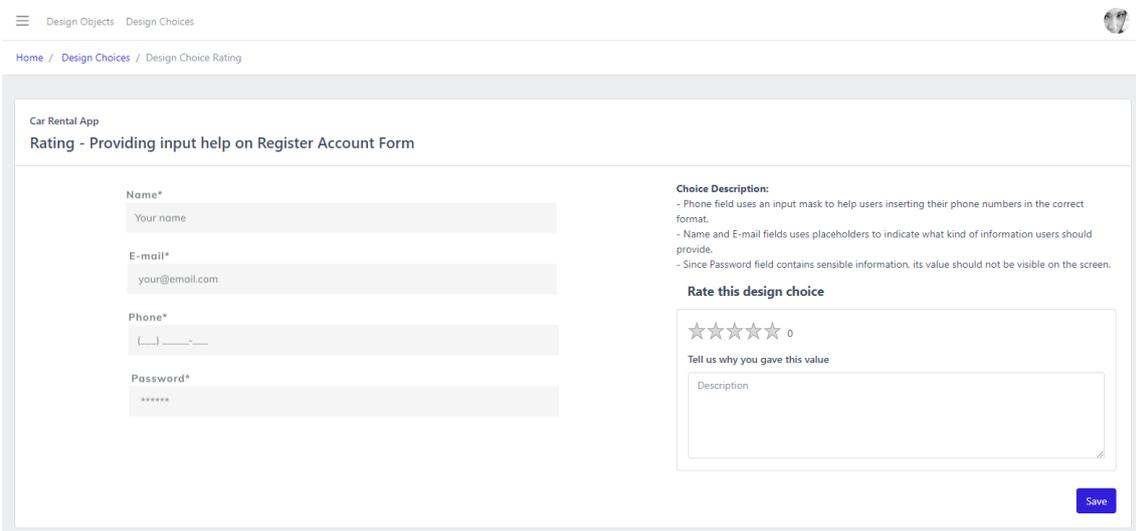


Figure 6.11 – Evaluating a design choice.

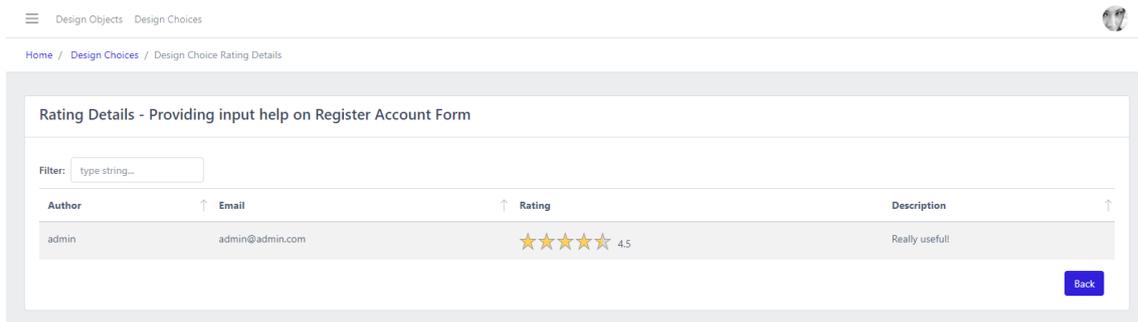


Figure 6.12 – Previous evaluations of a design choice.

The idea of developing KTID was influenced by the experience of the author of this dissertation with commercial tools (e.g., *Zeplin*⁵, *Avocode*⁶ and *Invision*⁷), when working on the development of interactive systems as a software engineer together with HCI designers. These tools were useful for communication and collaboration purposes, i.e., presenting design specifications created by HCI designers to other stakeholders and allowing them to make comments with feedback about the specifications. However, they did not provide a structured way of representing, storing and retrieving important pieces of information associated with design choices (e.g., design components) that were not explicitly described in design specifications (i.e., these pieces of information were clear in designers' minds, but were not accurately represented in artifacts, hampering a complete and correct understanding of them by other stakeholders). Although KTID is focused on supporting KM aspects, we took the experience with these tools as an inspiration for the way of using KTID, in the sense of storing and making available design specifications created in other tools. By using commercial tools as a reference, we aimed to propose a tool able to meet the needs of HCI design professionals and, consequently, reduce the gap between academy and industry identified in our previous investigation about KM in HCI design (studies presented in chapters 3 and 4).

It is important to highlight that KTID cannot be considered a complete KM solution or a knowledge management system, since it does not provide support to all activities in the KM cycle (e.g., it does not provide a robust curation to assess and make available knowledge items considered valuable for the organization). Table 6.1 associates KTID use cases with the activities that they support in the integrated KM cycle (DALKIR, 2011). We decided to leave out more robust KM features aiming to make a simpler KM solution, requiring minimal additional effort to integrate it into HCI design process. As we pointed in Chapter 4, the use

⁵ <https://zeplin.io/>

⁶ <https://avocode.com/>

⁷ <https://www.invisionapp.com/>

of lightweight technologies and a divide and conquer strategy to reduce complexity of the adoption of a KM approach might be useful to take more KM solutions for HCI design into industrial environments, also contributing to reduce the gap between theory and practice.

Table 6.1 – Support of KTID use cases to KM activities.

KM Activities	KTID Use Cases
Knowledge capture and/or creation	Manage Design Objects, Manage User Requirements, Manage Design Specifications, Describe Design Choices
Knowledge sharing and dissemination	Search for Design Choices
Knowledge acquisition and application	Evaluate Design Choices

Thus, the use of HCIDO as a basis to develop KTID conceptual model served as a proof of concept that showed that it is feasible to use HCIDO to develop KM solutions as the one built in this work. As a proof of concept, its results only indicate that using HCIDO is viable, and it does not mean that HCIDO is useful in practical settings. Therefore, studies about the use of HCIDO by other people than the researchers involved in this work are necessary.

6.3 Evaluating KTID

Aiming to evaluate KTID and, consequently, the feasibility of applying HCIDO to support knowledge management solutions in HCI design, we carried a study in which two HCI designers used the tool in an HCI design scenario. With this study, we sought to find preliminary evidence that allows us to evaluate and improve the feasibility of using the tool, as well as its utility.

6.3.1 Study Planning

The study **goal** was to evaluate if KTID is useful to support knowledge management aspects in HCI design and if its use is feasible. Following the GQM approach (BASILI; CALDIERA; ROMBACH, 1994), this goal is formalized as follows: *Analyze* KTID, *for the purpose of* evaluating its use to aid in HCI design, *with respect to* the utility and the feasibility of using it for knowledge representation, storage, retrieval and assessment, *from the viewpoint of* HCI designers, *in the context of* the development of interactive systems. In order to analyze the results, the following indicators were considered: utility and feasibility. The former was evaluated considering the participants perceptions about the adequacy of the tool (taking its

purpose into account) and how much the tool helped them in HCI design. The latter considered the participants' perceptions about ease of use and how much feasible they considered using the tool to aid in HCI design. Benefits and drawbacks pointed by the participants were also considered to indicate if the tool is useful and feasible.

The **instrument** used in the study consisted of four artifacts: (i) a document presenting the context of this work, the main functionalities of KTID and the instructions to be considered by the participants when using the tool; (ii) a consent form to participate in the study, which aims to safeguard the participants' rights regarding the study and its results; (iii) a form to characterize the participants' profile, which aims to obtain information about the participants' knowledge and experience in HCI design; and (iv) a questionnaire that allows participants to record their perception after using the tool. The forms were prepared with the help of Google Forms and are presented in Appendix A of this dissertation together with the document cited in (i).

The **procedure** adopted in the study consisted in inviting the participants and sending them the document that presents KTID and the study instructions. The author of this dissertation also made a brief presentation of the document and made himself available to assist and answer questions during the study. The **participants** were chosen considering convenience criteria (i.e., they were invited based on the researchers' relationship network and should be available to participate within the time necessary to conclude this dissertation). Four people were invited and only two of them were available to participate in the study. After the KTID presentation made by the author of this work, the participants should use KTID to support them in performing the HCI design activity described in the document and then answer a questionnaire in which they should record their perception about the use of KTID in supporting HCI design.

The **questionnaire** included questions aimed to extract the participants' perception about the adequacy of KTID in supporting KM aspects in HCI design, the utility of the tool for HCI design and the feasibility of using it in HCI design practice. The questionnaire consisted of objective and subjective questions. For the objective ones, the participants were asked to justify their answers. There was also a subjective question in which the participants could provide general improvement suggestions to KTID, aiming to provide a better support to KM aspects in HCI design. A fragment of the questionnaire is presented in Figure 6.13.

Feedback Questionnaire

1. Did you find out in the tool relevant knowledge to produce the proposed design solution? *

Yes

No

*
Justification:

Your answer _____

2. Did you include in the tool knowledge about the solution produced by you that may be relevant to produce possible future solutions? *

Yes

No

*
Justification:

Your answer _____

Figure 6.13 – Fragment of the feedback questionnaire about the use of KTID.

6.3.2 Study Execution

The participants of the study were two designers who had medium HCI design theoretical knowledge (i.e., acquired mainly during courses or undergraduate research) and low KM theoretical knowledge (i.e., acquired by themselves through books, videos or other materials). They reported they had already used KM resources to support HCI design on their own, without following organizational practices. Participant P1 was an undergraduate student with low practical experience in HCI design (i.e., less than one year) and participant P2 had bachelor’s degree and a high practical experience in HCI design (i.e., more than three years).

Following the planned procedure, after a presentation made by the researcher, the participants read the document with the instructions and the description of an HCI design scenario for which they should produce a design solution (i.e., an HCI Design Specification in the sense of HCIDO). The scenario description and the instructions are presented in

Appendix A. Before the study, we recorded an initial data set in KTID, aiming to help the participants understand which kind of information they could manage in the tool.

During the study execution, participant P1 asked some questions regarding the use of the tool and uploaded to KTID only the mobile version of the design specification for the interactive system proposed in the instructions document. Participant P2 did not make questions during the study execution and did not produce any design specification, justifying that he could understand, use and evaluate the tool by inserting fictitious data.

After the participants use KTID and answer the feedback questionnaire, the author carried a brief interview with the participants aiming to capture their overall perception about the tool and validate their answers to the questionnaire.

6.3.3 Study Results

This section summarizes the answers and the comments provided by the participants to each of the questions of the questionnaire.

a) Use of KTID to find relevant previous knowledge to support HCI design:

Both participants reported that they found relevant knowledge in KTID to help them produce design solutions. Participant P1 commented that the available design choices served as inspiration and supported her to make new design choices according to what she was supposed to design in the study. Participant P2, in turn, highlighted that, although he found useful knowledge available in the tool, some knowledge items were confusing.

b) Use of KTID to include new knowledge to support HCI design in the future: Only participant P1 included new knowledge in the tool. She justified her answer with the fact that she usually considers references from design choices of other interactive systems and makes her design choices based on what is adequate in the context for which she is designing. Since she is benefited from design choices from others, it also makes sense for her to include design choices her own. On the other hand, participant P2 did not include new knowledge because he informed that he could quickly understand and evaluate the tool by only reading the instructions and inserting some fictitious data, since he uses similar tools in his daily routine as a designer.

c) KTID's adequacy for supporting KM activities in HCI design: Both participants considered KTID adequate at a certain point for supporting the following KM activities: knowledge representation, knowledge storage, knowledge retrieval and knowledge evaluation. Participant P1 considered that KTID is adequate for supporting all those KM activities and complemented that she considered the tool promising for storing design

systems aiming to guarantee that designers follow the same guidelines and visual identity in a particular project or organization. Participant P2, in turn, considered that KTID is partially adequate for supporting those KM activities and explained that he considered using the tool a good idea, however it still cannot be used in an optimal format (i.e., in his opinion, the tool could be more practical and visual, since in most situations he needs to quickly consult knowledge displayed in a visual way).

d) KTID's utility in HCI design: Participant P1 considered the tool useful for aiding in HCI design, for the same reasons she considered it adequate for supporting KM activities. She also complemented that the use of the tool could facilitate communication in general and reduce the time spent in designing. On the other hand, participant P2 considered the tool neutral in terms of its utility. He commented that the tool lacks a “more real-world dynamics”, which should consider different professional profiles of designers (i.e., users of the tool). Furthermore, he also considered that formally specifying aesthetic aspects in the tool is not as useful as specifying functional aspects, because aesthetic aspects are more connected to personal choices and are more likely to change when they do not comply with the purpose of the system or with users or customers opinions.

e) Results obtained from the use of KTID in HCI design: The opinion of the participants differed regarding the results that can be obtained from using KTID in HCI design. Participant P1 believed that KTID can contribute to increase the quality of the HCI design solution, to reduce the effort designing the solution and to spend less time designing the solution. She also said that reusing previous design choices can contribute to the creative process and to inspire new design choices. Participant P2, in turn considered that KTID can partially help reach those results, because he is familiar with using similar tools in his daily routine, and he probably could only evaluate the actual contribution of KTID by using it in a real-world scenario.

f) KTID's ease of use in HCI design: None of the participants considered KTID easy to use. Participant P1 had a neutral perception regarding the tool's ease of use and reported that she missed some elements that could improve the navigability and usability of the tool, such as “next” and “previous” buttons in design choices visualization (instead of needing to return to the search page), and more complete edit features for design specifications and design choices. In participant P2's opinion, the tool was hard to use and lacked the exploration of visual aspects, since designers are used to think in visual terms (i.e., text descriptions should be displayed in a secondary moment to provide a deeper understanding about the content).

g) Feasibility of using KTID in HCI design: The participants considered KTID feasible to be used in supporting HCI design. In participant P1's opinion the use of the tool is very feasible, because of the benefits it brought to HCI design, by allowing designers to store and retrieve ideas. Participant P2, in turn, considered KTID just feasible to be used, as long as it is improved in terms of navigation flow, user interface and user interaction aspects.

h) Recommendation of the use of KTID to other people: Only participant P1 would recommend other people to use KTID, because she took into account her own experience with the tool and considered it interesting and promising. Participant P2 would not recommend the tool in its current stage of development because he considered that it is not mature enough to be used in HCI design practice.

i) Benefits that could be obtained and difficulties that could be faced when using KTID in a practical HCI design context: Both participants reported that they could obtain benefits from using KTID in a practical HCI design context. The benefits reported by participant P1 are storing design standards and their application context. Participant P2 informed that the tool can provide more quality control, standardization and agility in developing new products, but it still needs to be improved. Difficulties were only reported by participant P2. He believed that using the tool could probably stiffen the process of emerging new and different ideas (i.e., the reuse of design choices could lead to the repetition of the same solutions).

j) Suggestions for the evolution of KTID in order to improve the support for KM aspects in assisting HCI design: The following suggestions were provided by the participants in order to improve KTID: (i) make the tool more user friendly and beautiful; (ii) provide more robust edit features to design choices and design specifications; (iii) include navigation buttons on design choices view; and (iv) design a new user flow and a new layout for the tool's user interface based on similar tools used by designers, such as *Trello*, *Zero Height* and *Pinterest*.

6.3.4 Discussion

In this section, we present a discussion about the results presented in the previous section in terms of the indicators defined on the study planning. Results from questions (a) to (e) were used to analyze KTID's usefulness. In this context, questions (a) to (c) refer particularly to adequacy. Questions (f), (g) and (h) were used to analyze KTID in terms of the feasibility of using the tool in HCI design. Finally, question (i) provided results to analyze

KTID in terms of both usefulness and feasibility (i.e., if there are several benefits and few difficulties in using KTID, it means that the tool is feasible and useful).

Concerning the adequacy of KTID for supporting KM aspects in HCI design, the results from the study indicated that the tool may be *adequate but should be improved* to allow the representation of HCI design knowledge using more visual elements (e.g., images). As HCI designers deal with design objects that, in general, are most perceived in visual terms by users, it is natural that HCI designers also employ more their visual sense when acquiring new HCI design knowledge. We believe that replacing the table view of KTID design choices search with a more fluid layout that highlights image elements could contribute to improve KTID's adequacy in supporting KM aspects in HCI design.

We observed that, in general, participant P1 considered the tool more useful than participant P2. Considering the difference in their experience in HCI design practice, this result may indicate that *KTID can be more useful for novice HCI designers than to expert HCI designers*. This is an expected result, since expert designers rely on a larger amount of tacit knowledge which they have acquired during their HCI design practice. On the other hand, novice designers with less experience in HCI design may need more support from formalized knowledge. Therefore, we believe that a future improvement of KTID should address expert and novice designers in two different user profiles with different needs and use cases. While the former should be more focused on representing and storing knowledge, the latter should be more target to retrieving and reusing knowledge.

KTID was viewed by the participants as a *promising tool, but not easy to use*. Considering its current stage of development, its use is not appropriate in the industry yet. The use of *KTID was feasible in an experimental context* and can be feasible in an industrial context by addressing the participants suggestions and carrying further studies to improve the tool.

In summary, considering that the number of potential benefits of using KTID in HCI design is higher when compared with the number of difficulties, we can conclude that there is an indication that KTID can be useful and its use feasible. However, as pointed out before, the tool needs improvements.

Using KTID to aid in HCI design can contribute to standardization and, consequently, to increase quality, reduce effort and reduce the time spent in the development of new products. On the other hand, it can hamper creativity, which could be a side effect from the standardization benefit. Since the promotion of standardization ensures consistency in the design, it also reduces the number of possible design choices that can be made. To avoid this limitation, we suggest using KTID as a complementary source of knowledge in

HCI design combined with other external sources of knowledge to promote a mix between knowledge reuse and creation.

As for the future evolution of KTID, based on participants perception we believe that substantial improvements can be obtained by enhancing presentation and interaction aspects of the tool. Hence, we suggest for future studies the implementation and the evaluation of a new user interface in KTID, which can be inspired by commercial tools that are already familiar to HCI designers. By addressing more HCI aspects in KTID development, the tool can come closer to meeting the needs of HCI design professionals, increasing the chance of using it in practice.

The overall results of the study indicated that KTID aided in HCI design, although it needs to be improved in terms of user interface and interaction aspects. Consequently, the use of HCIDO may be suitable for supporting KM solutions in HCI design. Considering that the use of HCIDO in KTID development occurred in its conceptual phase and that KTID was evaluated from its users' point of view, further studies are necessary to evaluate the application of HCIDO to aid in conceptual modeling and development of HCI design tools.

6.3.5 Threats to Validity

As any study, KTID evaluation study has some limitations that may have threatened the validity of its results. Thus, these limitations must be considered together with the results. In this section we discuss some threats involved in the study.

One limitation to be considered is the short deadline that was given to participants perform the study. A few days were made available because we needed to have enough time to analyze the study results and to conclude this dissertation. Moreover, the study was carried remotely, thus the participants may have performed other tasks parallel to the study. If the participants had more time and if we could guarantee that they were exclusively focused on the study, maybe the study results could have been different.

The behavior of each participant when performing the study activities was different, which may also have threatened the results. Participant P1 produced a design solution while using KTID and P2 only inserted fictitious data to evaluate the tool (even after we have asked him to follow the procedure described in the instructions document). Since our goal was to evaluate the use of KTID, rather than evaluating the design solutions produced with the support of the tool, the impact of the lack of a design solution produced by participant P2 can be minimized. However, it is still a threat, since, different from P1, P2 did not use

the tool to develop an HCI design. Moreover, the different participants' behaviors were even useful for us to notice that depending on the experience level of the designer, he/she can use the tool under a different perspective and with different purposes.

With regard to the quality of the answers provided by the participants, there is the threat of the participants having misunderstood some questions. To address this threat, the author of this dissertation made himself available to answer questions and support the participants while they carried the study. The participants may also have not understood the tool's use cases or the motivation for using it. To mitigate this threat, we made a brief presentation about the tool and provided a document describing the tool's main use cases and the context in which it was developed. The questions contained in the questionnaire can also be a threat to the results. Some of them can lead to confirmation bias. We minimized this threat by asking the participants to justify their answers, so that they could reflect about the given answers instead of only answer yes or no.

Another limitation refers to the fact that the study occurred in a controlled environment that do not necessarily reflect a real-world scenario. Moreover, the participants used the tool in a short period of time. Hence, further studies are necessary to evaluate the use of KTID in industrial contexts. The small number of participants is also a meaningful threat to the results. Moreover, the participants were invited based on the researchers' relationship network, which may have influenced the answers.

Considering these threats, the study results cannot be generalized and must be understood as preliminary evidence that KTID, a KM solution built based on HCIDO, can be useful and feasible to support HCI design.

6.4 Concluding Remarks

This chapter presented the Knowledge Tool for Interaction Design (KTID), a computational tool developed based on HCIDO to support KM aspects in HCI design. HCIDO conceptualization was used to produce the conceptual model of KTID, reducing the effort in the conceptual modeling and providing knowledge about the application domain. The tool was developed to meet the requirement R7 (the ontology must be used to solve problems), established to HCIDO and presented in Chapter 1.

KTID was evaluated in a study which analyzed the utility and feasibility of using KTID to support knowledge management aspects in HCI design. The study indicated that the tool may be useful and its use may be feasible, as required in the evaluation criteria C3 (the solution built based on the ontology must be feasible and useful) established to HCIDO

and presented in Chapter 1. However, the tool must be improved in terms of user interface and interaction aspects to become more user friendly. The results of KTID evaluation also suggest that HCIDO can be applied to support the development of KM solutions as the one built in this work. The performed study has several limitations, thus the results should be considered as initial evidence and must be complemented by further studies.

Chapter 7

Final Considerations and Future Work

This chapter presents the final considerations (Section 7.1), contributions (Section 7.2) and proposals of future works (Section 7.3) to continue and improve the work proposed in this dissertation.

7.1 Final Considerations

HCI design plays an important role in the development of interactive systems, since it is concerned with how the system should be designed to support users achieving their goals through the interaction with the system (SUTCLIFFE, 2014). However, there have been communication and knowledge transfer challenges in the integration of HCI design knowledge, principles and methods into SE processes, since HCI design involves a diverse body of knowledge from several fields and embodies a large amount of tacit knowledge (BOFYLATOS; SPYROU, 2017; CARROLL, 2014). Moreover, due to the knowledge intersection between HCI and SE, different meanings can be associated in each area to the same term, which can lead to semantic interoperability issues (OGUNYEMI; LAMAS, 2014).

In this sense, ontologies can help to capture and organize knowledge about HCI design based on a common vocabulary to deal with semantic interoperability and knowledge-related problems (STUDER; BENJAMINS; FENSEL, 1998). They can be used to support KM technologies to provide knowledge access, optimize knowledge retrieval, support communication mechanisms and, therefore, knowledge exchange (VARMA, 2007). However, our investigation about knowledge management in HCI design indicated that ontologies have not been much used in KM solutions, although the lack of a common conceptualization about HCI design has been one of the main challenges reported both in the literature and by HCI design practitioners.

Considering this scenario, in this work, we explored the combination of ontologies and ontology networks with KM to potentialize knowledge creation, transfer and reuse in the context of the HCI design of interactive systems. Hence, the main objective of this work was to propose a well-founded consensual conceptualization of HCI design to support knowledge management solutions to aid in HCI design of interactive systems. This main objective was detailed in four specific objectives, and all of them were achieved. Table 7.1

presents the specific objectives of this work and the main product that serves as evidence that the objective was achieved.

Table 7.1 – Specific objectives of this work.

Objectives	Products
Investigate the state of the art about knowledge management in HCI design.	Systematic Mapping (Chapter 3)
Investigate the state of the practice about knowledge management in HCI design.	Survey (Chapter 4)
Develop a reference ontology about HCI design of interactive systems.	HCIDO and SDRO (Chapter 5)
Apply the reference ontology to support HCI design of interactive systems.	KTID (Chapter 6)

Among the limitations of this work, we can highlight its evaluation. KTID was evaluated by a limited number of professionals, in a noticeably short period and outside the organizational context. Moreover, the evaluation of the participants may have been influenced by limitations of the tool (which was not the focus of this work) rather than to the ontology. Hence, the results of the evaluation cannot be considered conclusive.

The tool needs to be improved to be able to achieve our goal of getting academic research results closer to practical settings. Due to time constraints, it was not possible to develop a tool suitable for being delivered to the Industry. However, we must emphasize that, in this work, the tool is a means to apply the proposed ontology, which is the main artifact produced in this work. The ontology itself also needs further evaluation, being required to be used for other people to solve other knowledge-related problems.

Furthermore, it is important to point out that HCI is a multidisciplinary area that deals with human aspects, thus the problem addressed in this work may also be influenced by social, cultural, psychological and other factors. Therefore, we believe that the combination of the use of ontologies and KM solutions as the ones built in this work can contribute to solve communication and knowledge transfer issues in HCI design, but it should not be used as the only approach to handle the problem.

7.2 Contributions

The main contributions of this work are:

- (i) The *HCI Design Ontology* (HCIDO), which provided a well-founded conceptualization about HCI design in the context of interactive systems development.
- (ii) The *Software Design Reference Ontology* (SDRO), which provided a well-founded conceptualization about design in the software context considering its mental and physical natures and was reused in HCIDO.
- (iii) The proposal of *KTID*, the computational tool based on HCIDO and developed in (OGIONI, 2021), which demonstrated that the ontology can be applied in practice to support knowledge management aspects in HCI design.
- (iv) The *systematic mapping*, which investigated the state of the art concerning the use of KM in the HCI design context and provided a panorama of research related to the topic. The systematic mapping main results were published in (CASTRO *et al.*, 2020). The paper received the best paper award of the Experimental Software Engineering Track of CIBSE 2020 and an extended version is currently under review in the Journal of Software Engineering Research and Development.
- (v) The *survey*, which investigated KM in HCI design practice and complemented the systematic mapping, identifying gaps and improvement opportunities to organizations interested in applying KM initiatives in HCI design context.

7.3 Future Work

Considering the current stage presented here, some perspectives for future work are presented below. Concerning the *research* scope, we can highlight:

- (i) Update the investigation in the literature, to verify if new works have been published reporting the use of KM in HCI design. Moreover, aiming to provide information to improve the use of KM in HCI design, the results obtained in our mapping study can be compared with results from other studies investigating KM use in other domains (e.g., requirements engineering) and KM solutions proposed in other domains can inspire new proposals to support HCI design by using KM.

- (ii) Extend the survey with HCI design practitioners to include more participants from different countries and to further investigate other aspects of KM in HCI design (e.g., which knowledge management activities HCI designers have been performing).
- (iii) Use the ontologies proposed in this work to support other types of applications (e.g., semantic documentation or semantic interoperability between tools).
- (iv) Explore the use of HCIDO integrated to other ontologies of SEON and HCI-ON, to provide more comprehensive solutions (e.g., covering from user requirements to testing; integrating HCI design and HCI evaluation).
- (v) Propose a KM process associated with a complete KM solution to support HCI design, in order to extend the coverage of the solution proposed in this work.

In relation to the *well-founded conceptualization of HCIDO and SDRO*:

- (i) Define and formalize axioms to address constraints that are not captured in the conceptual models of SDRO and HCIDO. This is a very important issue that was not addressed in this work due to time limitation.
- (ii) Extend the scope of both ontologies, addressing more types of design components and design specifications, as well as detailing with other mental aspects that motivate design choices, such as standards, best practices, intuition, among others. In HCIDO, the design of interactive computer systems (including hardware aspects) can also be addressed.
- (iii) Continue the development process for HCIDO and SDRO, creating operational ontologies that could be used to support HCI in runtime, improving visualization capabilities (e.g., supporting information clustering and adaption of user interface appearance) or improving interaction possibilities (e.g., supporting input assistance and user interface integration) (PAULHEIM; PROBST, 2010).
- (iv) Extend the validation of the ontologies by using formal validation techniques (e.g., using Alloy).
- (v) Extend the evaluation of the ontologies by instantiating other scenarios (e.g., real-world situations).

- (vi) Perform an evaluation study to evaluate the use of HCIDO regarding its capacity of supporting the conceptual modeling of HCI design tools.
- (vii) Integrate SDRO with the Design Process Ontology (DPO) and develop a new HCI design process ontology integrated to HCIDO, providing a conceptualization that addresses design both as a noun and as a verb.

Concerning *KTID: the computational tool to support knowledge in HCI design*:

- (i) Integrate an operational version of HCIDO to the tool, in order to enhance some of its functionalities (e.g., making complex search queries using SPARQL).
- (ii) Carry out new evaluations of the tool, with more HCI designers and organizations and considering real projects. Then, use the results to make improvements in the tool, aiming to increase the chance of using it in practice.
- (iii) Enhance the identification of design choices and components in the tool with text and image recognition combined with artificial intelligence, suggesting possible candidates based on past records, which could reduce the human effort for inserting new knowledge in the tool.
- (iv) Enhance the reuse of design choices with an intelligent search mechanism that suggests previous design choices motivated by similar requirements or design choices of a current project.
- (v) Integrate the tool with other HCI design tools, using HCIDO as a reference framework for aligning their concepts and data models.
- (vi) Implement the improvements suggested by the participants of KTID evaluation study, making the tool easier to use by HCI designers.

References

BAKAEV, M.; GAEDKE, M. Application of evolutionary algorithms in interaction design: From requirements and ontology to optimized web interface. *In: NW RUSSIA YOUNG RESEARCHERS IN ELECTRICAL AND ELECTRONIC ENG. CONF. 2016, Anais [...]. [s.l: s.n.] p. 129–134.*

BAKAEV, Maxim; AVDEENKO, Tatiana. Ontology to Support Web Design Activities in E-Commerce Software Development Process. *[S. l.]*, n. August 2015, 2010. DOI: 10.2316/P.2010.691-075.

BAKER, Alex; HOEK, André Van der. **Examining Software Design from a General Design Perspective.** Irvine.

BASIL, V.; CALDIERA, G.; ROMBACH, H. D. The Goal Question Metric Approach. *In: 1994, Anais [...]. [s.l: s.n.]*

BECERRA-FERNANDEZ, Irma; SABHERWAL, Rajiv. **Knowledge Management: Systems and Processes.** New Delhi: PHI Learning Private Limited, 2010.

BENYON, D. Designing interactive systems : a comprehensive guide to HCI, UX and interaction design. *In: 2013, Anais [...]. [s.l: s.n.]*

BOFYLATOS, Spyros; SPYROU, Thomas. Meaning, knowledge and artifacts, giving a voice to tacit knowledge. **The Design Journal**, *[S. l.]*, v. 20, n. sup1, p. S4422–S4433, 2017. DOI: 10.1080/14606925.2017.1352938.

BOUWMEESTER, Niels. A Knowledge Management Tool for Speech Interfaces (Poster Abstract). *In: PROCEEDINGS OF THE 22ND ANNUAL INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL 1999*, New York, NY, USA. **Anais [...].** New York, NY, USA: ACM, 1999. p. 293–294. DOI: 10.1145/312624.312721.

BRANK, Janez; GROBELNIK, Marko; MLADENIĆ, Dunja. A survey of ontology evaluation techniques. *In: IN IN PROCEEDINGS OF THE CONFERENCE ON DATA MINING AND DATA WAREHOUSES (SIKDD 2005) 2005, Anais [...]. [s.l: s.n.]*

BRINGUENTE, Ana Christina de Oliveira; FALBO, Ricardo de Almeida; GUIZZARDI, Giancarlo. Using a Foundational Ontology for Reengineering a Software Process

Ontology. **Journal of Information and Data Management**, [S. l.], v. 2, n. 3, p. 511–526, 2011. DOI: 10.5753/jidm.2011.1424.

BUDGEN, D. **Software design**. 2nd ed ed. Harlow, England ; New York: Addison-Wesley, 2003.

CARROLL, John Millar. Human Computer Interaction (HCI). *In*: SOEGAARD, Mads; DAM, Rikke Friis (org.). **The Encyclopedia of Human-Computer Interaction**. 2nd. ed. Aarhus, Denmark: The Interaction Design Foundation, 2014. p. 21–61.

CASTRO, Murillo V. H. B.; COSTA, Simone Dornelas; BARCELLOS, Monalessa P.; FALBO, Ricardo de A. Knowledge management in human-computer interaction design: A mapping study. *In*: 23RD IBEROAMERICAN CONFERENCE ON SOFTWARE ENGINEERING, CIBSE 2020 2020, **Anais [...]**. [s.l: s.n.]

CHAMMAS, Adriana; QUARESMA, Manuela; MONT'ALVÃO, Cláudia. A Closer Look on the User Centred Design. **Procedia Manufacturing**, [S. l.], v. 3, p. 5397–5404, 2015. DOI: <https://doi.org/10.1016/j.promfg.2015.07.656>.

CHERA, C.; TSAI, W.; VATAVU, R. Gesture ontology for informing Service-oriented Architecture. *In*: INT. SYMPOSIUM ON INTELLIGENT CONTROL 2012, **Anais [...]**. : IEEE, 2012. p. 1184–1189.

CHEWAR, C. M.; BACHETTI, Edwin; MCCRICKARD, D. Scott; BOOKER, John E. Automating a Design Reuse Facility with Critical Parameters. *In*: (Robert J. K. Jacob, Quentin Limbourg, Jean Vanderdonckt, Org.)COMPUTER-AIDED DESIGN OF USER INTERFACES IV 2004, Dordrecht. **Anais [...]**. Dordrecht: Springer Netherlands, 2004. p. 235–246.

CHEWAR, C. M.; MCCRICKARD, D. S. Links for a Human-Centered Science of Design: Integrated Design Knowledge Environments for a Software Development Process. *In*: PROCEEDINGS OF THE 38TH ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES 2005, **Anais [...]**. [s.l: s.n.] p. 256c-256c. DOI: 10.1109/HICSS.2005.390.

COSTA, Simone Dornelas; BARCELLOS, Monalessa Perini; FALBO, Ricardo de Almeida; CASTRO, Murillo Vasconcelos Henriques Bittencourt. Towards an Ontology Network on Human-Computer Interaction. *In*: (Gillian Dobbie, Ulrich Frank, Gerti

Kappel, Stephen W. Liddle, Heinrich C. Mayr, Org.)CONCEPTUAL MODELING 2020, Cham. **Anais [...]**. Cham: Springer International Publishing, 2020. p. 331–341.

COSTA, Simone Dornellas. **An Ontology Network to support Knowledge Representation and Semantic Interoperability in the HCI Domain**. 2021. Federal University of Espírito Santo, Vitória – ES, Brazil., [S. l.], 2021.

D'AQUIN, Mathieu; GANGEMI, Aldo. Is there beauty in ontologies? **Applied Ontology**, [S. l.], v. 6, p. 165–175, 2011. DOI: 10.3233/AO-2011-0093.

DALKIR, K. **Knowledge Management in Theory and Practice, second edition**The MIT press Cambridge, Massachussets, , 2011.

DE MEDEIROS, Adriana Pereira; SCHWABE, Daniel; FEIJÓ, Bruno. Kuaba Ontology: Design Rationale Representation and Reuse in Model-Based Designs. *In*: (Lois Delcambre, Christian Kop, Heinrich C. Mayr, John Mylopoulos, Oscar Pastor, Org.)CONCEPTUAL MODELING – ER 2005 2005, Berlin, Heidelberg. **Anais [...]**. Berlin, Heidelberg: Springer, 2005. p. 241–255. DOI: 10.1007/11568322_16.

DE SOUZA, Clarisse Sieckenius. **The Semiotic Engineering of Human-Computer Interaction**. [s.l.] : The MIT Press, 2005.

DESIGN. **Design meaning in the Cambridge English Dictionary**, 2020. a.

DESIGN. **Definition of Design in Merriam-Webster Dictionary**, 2020. b.

DIX, Alan; DIX, Alan John; FINLAY, Janet; ABOARD, Gregory D.; BEALE, Russell. **Human-computer interaction**. [s.l.] : Pearson Education, 2003.

DUARTE, Bruno Borlini; DE CASTRO LEAL, Andre Luiz; FALBO, Ricardo De Almeida; GUIZZARDI, Giancarlo; GUIZZARDI, Renata S. S.; SILVA SOUZA, Vítor E. Ontological foundations for software requirements with a focus on requirements at runtime. **Applied Ontology**, [S. l.], v. 13, n. 2, p. 73–105, 2018. DOI: 10.3233/AO-180197.

FAIRCLOUGH, Stephen H. Fundamentals of physiological computing. **Interacting with computers**, [S. l.], v. 21, n. 1–2, p. 133–145, 2009.

FALBO, Ricardo de A. **Car Rental Software Design Specification v1.0**. 2018.

Disponível em: <http://www.inf.ufes.br/~jssalomon/wp->

content/uploads/disciplinas/projsistsoft/trabalho/Documento_Projeto_v1.0.pdf. Acesso em: 26 jul. 2021.

FALBO, Ricardo de Almeida. SABiO: Systematic Approach for Building Ontologies. *In*: CEUR WORKSHOP PROCEEDINGS 2014, Rio de Janeiro, Brazil. **Anais [...]**. Rio de Janeiro, Brazil: CEUR-WS.org, 2014.

FALBO, Ricardo de Almeida; BARCELLOS, Monalessa Perini; NARDI, Julio Cesar; GUIZZARDI, Giancarlo. Organizing ontology design patterns as ontology pattern languages. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, [S. l.], v. 7882 LNCS, p. 61–75, 2013. DOI: 10.1007/978-3-642-38288-8-5.

GABRIEL, Isaac J. An Expert System for Usability Evaluations of Business-to-Consumer E-Commerce Sites. *In*: PROCEEDINGS OF THE 6TH ANNUAL ISONEWORLD CONFERENCE, LAS VEGAS, NV 2007, **Anais [...]**. [s.l: s.n.]

GALLAGHER, Shaun. **How the body shapes the mind**. [s.l.] : Clarendon Press, 2006.

GERO, John S. Design Prototypes: A Knowledge Representation Schema for Design. **AI Magazine**, [S. l.], v. 11, n. 4, p. 26, 1990. DOI: 10.1609/aimag.v11i4.854.

GERO, John S.; KANNENGIESSER, Udo. The Function-Behaviour-Structure Ontology of Design. *In*: CHAKRABARTI, Amaresh; BLESSING, Lucienne T. M. (org.). **An Anthology of Theories and Models of Design: Philosophy, Approaches and Empirical Explorations**. London: Springer, 2014. p. 263–283. DOI: 10.1007/978-1-4471-6338-1_13.

GUARINO, Nicola. **Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy**. [s.l.] : IOS press, 1998. v. 46

GUARINO, Nicola. Artefactual Systems, Missing Components and Replaceability. *In*: FRANSSEN, Maarten; KROES, Peter; REYDON, Thomas A. C.; VERMAAS, Pieter E. (org.). **Artefact Kinds: Ontology and the Human-Made World**. Cham: Springer International Publishing, 2014. p. 191–206. DOI: 10.1007/978-3-319-00801-1_11.

GUARINO, Nicola; MELONE, Maria Rosaria Stufano. On the Ontological Status of Design Objects. *In*: (Francesca Alessandra Lisi, Stefano Borgo, Org.)AIDE@ AI* IA

2015, **Anais** [...]. : CEUR-WS.org, 2015. p. 27–32.

GUIZZARDI, Giancarlo. **Ontological foundations for structural conceptual models**. 2005. Telematica Instituut / CTIT, [S. l.], 2005.

GUIZZARDI, Giancarlo. On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. *In*: PROCEEDINGS OF THE 2007 CONFERENCE ON DATABASES AND INFORMATION SYSTEMS IV: SELECTED PAPERS FROM THE SEVENTH INTERNATIONAL BALTIC CONFERENCE DB\&IS'2006 2007, NLD. **Anais** [...]. NLD: IOS Press, 2007. p. 18–39.

GUIZZARDI, Giancarlo; FALBO, Ricardo de Almeida; GUIZZARDI, Renata. Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology. *In*: CIBSE 2008, **Anais** [...]. [s.l: s.n.]

GUIZZARDI, Giancarlo; WAGNER, Gerd; DE ALMEIDA FALBO, Ricardo; GUIZZARDI, Renata S. S.; ALMEIDA, João Paulo A. Towards ontological foundations for the conceptual modeling of events. *In*: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING 2013, Berlin, Heidelberg. **Anais** [...]. Berlin, Heidelberg: Springer, 2013. p. 327–341. DOI: 10.1007/978-3-642-41924-9_27.

HAPPEL, Hans-Jörg; KORTHAUS, Axel; SEEDORF, Stefan; TOMCZYK, Peter. KOnToR: An Ontology-enabled Approach to Software Reuse. *In*: 2006, **Anais** [...]. [s.l: s.n.] p. 349–354.

HARNING, Morten; VANDERDONCKT, Jean; FLORINS, Murielle. Closing the Gaps: Software Engineering and Human-Computer Interaction. [S. l.], 2003.

HARRISON, Steve; TATAR, Deborah; SENGERS, Phoebe. The three paradigms of HCI. *In*: ALT. CHI. SESSION AT THE SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS SAN JOSE, CALIFORNIA, USA 2007, **Anais** [...]. [s.l: s.n.] p. 1–18.

HEFLEY, William E.; BUIE, Elizabeth A.; LYNCH, Gene F.; MULLER, Michael J.; HOECKER, Douglas G.; CARTER, Jim; ROTH, J. Thomas. Integrating human factors with software engineering practices. *In*: PROCEEDINGS OF THE HUMAN FACTORS AND ERGONOMICS SOCIETY ANNUAL MEETING 1994, **Anais** [...]. [s.l: s.n.] p. 315–319.

HENNINGER, Scott; HAYNES, Kyle; REITH, Michael W. A Framework for Developing Experience-based Usability Guidelines. *In: PROCEEDINGS OF THE 1ST CONFERENCE ON DESIGNING INTERACTIVE SYSTEMS: PROCESSES, PRACTICES, METHODS, & TECHNIQUES 1995*, New York, NY, USA. **Anais [...]**. New York, NY, USA: ACM, 1995. p. 43–53. DOI: 10.1145/225434.225440.

HENNINGER, Scott; KESHK, Mohamed; KINWORTHY, Ryan. Capturing and Disseminating Usability Patterns with Semantic Web Technology. [*S. l.*], 2004.

HEVNER, Alan R. A three cycle view of design science research. **Scandinavian journal of information systems**, [*S. l.*], v. 19, n. 2, p. 4, 2007.

HEWETT, Thomas T.; BAECKER, Ronald; CARD, Stuart; CAREY, Tom; GASEN, Jean; MANTEI, Marilyn; PERLMAN, Gary; STRONG, Gary; VERPLANK, William. **ACM SIGCHI curricula for human-computer interaction**. [s.l.] : ACM, 1992.

HUGHES, Michael. A Pattern Language Approach to Usability Knowledge Management. **J. Usability Studies**, Bloomingdale, IL, v. 1, n. 2, p. 76–90, 2006.

IEEE. **IEEE Std 1016-2009 (Revision of IEEE Std 1016-1998), IEEE Standard for Information Technology—Systems Design—Software Design Descriptions**. [s.l.: s.n.]. v. 2009

ISO/IEC/IEEE. **ISO/IEC/IEEE 24765 - Int. Standard - Systems and software engineering Vocabulary**, 2017.

ISO. **ISO 9241-210:2019(en) - Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems**Int. Organization for Standardization, 2019.

KITCHENHAM, Barbara A.; CHARTERS, Stuart. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [s.l.: s.n.].

KOHLHASE, Andrea E.; KOHLHASE, Michael. Semantic Transparency in User Assistance Systems. *In: PROCEEDINGS OF THE 27TH ACM INTERNATIONAL CONFERENCE ON DESIGN OF COMMUNICATION 2009*, New York, NY, USA. **Anais [...]**. New York, NY, USA: Association for Computing Machinery, 2009. p. 89–96. DOI: 10.1145/1621995.1622013.

KROL, Laurens R.; FREYTAG, Sarah-Christin; FLECK, Markus; GRAMANN, Klaus; ZANDER, Thorsten O. A task-independent workload classifier for neuroadaptive technology: Preliminary data. *In: 2016 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC) 2016*, **Anais [...]**. [s.l: s.n.] p. 3171–3174.

KULTSOVA, Marina; POTSELUICO, Anastasiya; ZHUKOVA, Irina; SKORIKOV, Alexander; ROMANENKO, Roman. A Two-Phase Method of User Interface Adaptation for People with Special Needs. *In: CREATIVITY IN INTELLIGENT TECH AND DATA SCIENCE*. 2017, **Anais [...]**. : Springer, 2017. p. 805–821.

MAFRA, Sômulo Nogueira; TRAVASSOS, Guilherme Horta. Estudos Primários e Secundários apoiando a busca por Evidência em Engenharia de Software. **Relatório Técnico, RT-ES**, [S. l.], v. 687, n. 06, 2006.

MCPHEE, Kent. **Design Theory and Software Design**. [s.l: s.n.]. DOI: 10.7939/R3MS3K15D.

MOHAMED, Mona A.; CHAKRABORTY, Joyram; DEHLINGER, Josh. Trading off Usability and Security in User Interface Design Through Mental Models. **Behav. Inf. Technol.**, Bristol, PA, USA, v. 36, n. 5, p. 493–516, 2017. DOI: 10.1080/0144929X.2016.1262897.

MYRGIOTI, Eirini; BASSILIADES, Nick; MILIOU, Amalia. Bridging the HASM: An OWL ontology for modeling the information pathways in haptic interfaces software. **Expert Systems with Applications**, [S. l.], v. 40, n. 4, p. 1358–1371, 2013.

NETO, Elias Harmuch; VAN AMSTEL, Frederick M. C.; BINDER, Fabio Vinicius; REINEHR, Sheila dos S.; MALUCELLI, Andreia. Trajectory and traits of devigners: a qualitative study about transdisciplinarity in a software studio. *In: 2020 IEEE 32ND CONFERENCE ON SOFTWARE ENGINEERING EDUCATION AND TRAINING (CSEE&T) 2020*, **Anais [...]**. [s.l: s.n.] p. 1–9.

NONAKA, Ikujiro. A dynamic theory of organizational knowledge creation. **Organization science**, [S. l.], v. 5, n. 1, p. 14–37, 1994.

NORMAN, Donald A. **The Design of Everyday Things: Revised and Expanded Edition**. 2nd. ed. [s.l.] : Basic Books, 2013.

O'LEARY, Daniel E. Enterprise Knowledge Management. **Computer**, Los Alamitos, CA,

USA, v. 31, n. 3, p. 54–61, 1998. DOI: 10.1109/2.660190.

OGIONI, Beatriz Sessa. **Uma Ferramenta Computacional para Apoiar Aspectos de Gerência de Conhecimento no Design de IHC**. 2021. Universidade Federal do Espírito Santo, Vitória-ES, Brasil, [S. l.], 2021.

OGUNYEMI, Abiodun; LAMAS, David. Interplay between human-computer interaction and software engineering. **Iberian Conference on Information Systems and Technologies, CISTI**, [S. l.], 2014. DOI: 10.1109/CISTI.2014.6877024.

OSTERWEIL, Leon J. A Future for Software Engineering? *In*: FUTURE OF SOFTWARE ENGINEERING (FOSE '07) 2007, **Anais [...]**. [s.l: s.n.] p. 1–11. DOI: 10.1109/FOSE.2007.1.

PAULHEIM, Heiko; PROBST, Florian. Ontology-Enhanced User Interfaces: A Survey. **Int. J. Semantic Web Inf. Syst.**, [S. l.], v. 6, p. 36–59, 2010. DOI: 10.4018/jswis.2010040103.

PAULHEIM, Heiko; PROBST, Florian. UI2Ont—A Formal Ontology on User Interfaces and Interactions. *In*: [s.l: s.n.]. p. 1–24. DOI: 10.1007/978-1-4471-5301-6_1.

PEFFERS, Ken; TUUNANEN, Tuure; ROTHENBERGER, Marcus A.; CHATTERJEE, Samir. A design science research methodology for information systems research. **Journal of management information systems**, [S. l.], v. 24, n. 3, p. 45–77, 2007.

PETERSEN, Kai; VAKKALANKA, Sairam; KUZNIARZ, Ludwik. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, Newton, MA, USA, v. 64, p. 1–18, 2015. DOI: <https://doi.org/10.1016/j.infsof.2015.03.007>.

PFLEEGER, Shari Lawrence. Design and analysis in software engineering: the language of case studies and formal experiments. **ACM SIGSOFT Software Engineering Notes**, [S. l.], v. 19, n. 4, p. 16–20, 1994.

POLANYI, Michael. **The Tacit Dimension**. Garden City, NY: Doubleday, 1966.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Software engineering: a practitioners approach**. 9th. ed. [s.l.] : McGraw Hill, 2020.

RALPH, Paul. The Sensemaking-Coevolution-Implementation Theory of software design.

Science of Computer Programming, Towards general theories of software engineering. [*S. l.*], v. 101, Towards general theories of software engineering, p. 21–41, 2015. DOI: 10.1016/j.scico.2014.11.007.

RALPH, Paul; WAND, Yair. A Proposal for a Formal Definition of the Design Concept. *In: DESIGN REQUIREMENTS ENG.: A TEN-YEAR PERSPECTIVE 2009*, **Anais [...]**. : Springer, 2009. p. 103–136.

ROGERS, Yvonne; SHARP, Helen; PREECE, Jenny. **Interaction Design: Beyond Human-Computer Interaction**. 3rd. ed. Chichester, United Kingdom: John Wiley & Sons, 2011.

RUS, I.; LINDVALL, M. Knowledge management in software engineering. **IEEE Software**, [*S. l.*], v. 19, n. 3, p. 26–38, 2002. DOI: 10.1109/MS.2002.1003450.

RUY, Fabiano Borges; FALBO, Ricardo de Almeida; BARCELLOS, Monalessa Perini; COSTA, Simone Dornelas; GUIZZARDI, Giancarlo. SEON: A Software Engineering Ontology Network. *In: KNOWLEDGE ENG. AND KNOWLEDGE MANAGEMENT 2016*, **Anais [...]**. : Springer, 2016. p. 527–542.

SAIYD, Nedhal Al; SAID, Intisar Al; NEAIMI, Afaf Al. Towards an ontological concepts for domain-driven software design. *In: 2009 FIRST INTERNATIONAL CONFERENCE ON NETWORKED DIGITAL TECHNOLOGIES 2009*, **Anais [...]**. [s.l: s.n.] p. 127–131. DOI: 10.1109/NDT.2009.5272092.

SCHERP, Ansgar; SAATHOFF, Carsten; FRANZ, Thomas; STAAB, Steffen. Designing core ontologies. **Applied Ontology**, [*S. l.*], v. 6, n. 3, p. 177–221, 2011.

SCHNEIDER, Kurt. **Experience and Knowledge Management in Software Engineering**. 1st. ed. Heidelberg, Berlin: Springer Publishing Company, Incorporated, 2009.

SCHÖN, Donald A. **The reflective practitioner: how professionals think in action**. New York: Basic Books, 1983.

SEFFAH, Ahmed; GULLIKSEN, Jan; DESMARAIS, Michel. **Human-Centered Software Engineering — Integrating Usability in the Software Development Lifecycle**. [s.l: s.n.]. DOI: 10.1007/1-4020-4113-6.

SIKORSKI, Marcin; GARNIK, Igor; LUDWISZEWSKI, Bohdan; WYRWIŃSKI, Jan. Knowledge Management Challenges in Collaborative Design of a Virtual Call Centre. *In: KNOWLEGE-BASED AND INTELLIGENT INFORMATION AND ENGINEERING SYSTEMS 2011*, Berlin, Heidelberg. **Anais [...]**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 657–666.

SILVA, Thiago Rocha; HAK, Jean Luc; WINCKLER, Marco. A Formal Ontology for Describing Interactive Behaviors and Supporting Automated Testing on User Interfaces. **International Journal of Semantic Computing**, [S. l.], v. 11, n. 4, p. 513–539, 2017. DOI: 10.1142/S1793351X17400219.

SIMON, Herbert A. **The sciences of the artificial**. 3rd ed ed. Cambridge, Mass: MIT Press, 1996.

SMITH, A.; DUNCKLEY, L. Prototype evaluation and redesign: structuring the design space through contextual techniques. **Interacting with Computers**, [S. l.], v. 14, n. 6, p. 821–843, 2002. DOI: 10.1016/S0953-5438(02)00031-0.

SMITH, J. L.; BOHNER, S. .. A.; MCCRICKARD, D. S. Toward introducing notification technology into distributed project teams. *In: 12TH IEEE INTERNATIONAL CONFERENCE AND WORKSHOPS ON THE ENGINEERING OF COMPUTER-BASED SYSTEMS (ECBS'05) 2005*, **Anais [...]**. [s.l.: s.n.] p. 349–356. DOI: 10.1109/ECBS.2005.69.

SOLANKI, Monika. DIO: A Pattern for Capturing the Intents Underlying Designs. *In: PROCEEDINGS OF THE 6TH WORKSHOP ON ONTOLOGY AND SEMANTIC WEB PATTERNS (WOP 2015) 2015*, **Anais [...]**. : CEUR-WS.org, 2015.

STAAB, Steffen; STUDER, Rudi (ORG.). **Handbook on Ontologies**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. DOI: 10.1007/978-3-540-24750-0.

STEPHANIDIS, Constantine; AKOUMIANAKIS, Demosthenes. Knowledge Management in HCI Design. *In: W. KARWOWSKI (org.). International Encyclopedia of Ergonomics and Human Factors*. Vol. 1 ed. [s.l.] : Taylor & Francis, 2001. p. 705–710.

STILL, Kaisa. Exploring Knowledge Processes in User-Centered Design Process. *In: THE 7TH EUROPEAN CONFERENCE ON KNOWLEDGE MANAGEMENT 2006*,

Anais [...]. [s.l.: s.n.] p. 533.

STUDER, Rudi; BENJAMINS, V. Richard; FENSEL, Dieter. Knowledge engineering: Principles and methods. **Data & Knowledge Engineering**, [S. l.], v. 25, n. 1, p. 161–197, 1998. DOI: 10.1016/S0169-023X(97)00056-6.

SUÁREZ-FIGUEROA, Mari Carmen; GÓMEZ-PÉREZ, Asunción; MOTTA, Enrico; GANGEMI, Aldo. **Ontology Engineering in a Networked World**. [s.l.] : Springer, 2012.

SUÁREZ, Pablo Ribeiro; JÚNIOR, Bernardo Lula; DE BARROS, Marcelo Alves. Applying knowledge management in UI design process. *In:* (Pavel Slavik, Philippe Palanque, Org.)PROCEEDINGS OF THE 3RD ANNUAL CONFERENCE ON TASK MODELS AND DIAGRAMS - TAMODIA '04 2004, NY, NY, USA. **Anais** [...]. NY, NY, USA: ACM Press, 2004. p. 113–120. DOI: 10.1145/1045446.1045468.

SUTCLIFFE, Alistair G. Requirements Engineering from an HCI Perspective. *In:* SOEGAARD, Mads; DAM, Rikke Friis (org.). **The Encyclopedia of Human-Computer Interaction**. 2. ed. Aarhus, Denmark: The Interaction Design Foundation, 2014. p. 707–760.

TAYLOR, Richard N.; VAN DER HOEK, Andre. Software Design and Architecture: The once and future focus of software engineering. *In:* FUTURE OF SOFTWARE ENGINEERING (FOSE '07) 2007, Minneapolis, MN. **Anais** [...]. Minneapolis, MN: IEEE, 2007. p. 226–243. DOI: 10.1109/FOSE.2007.21.

VALASKI, Joselaine; MALUCELLI, Andreia; REINEHR, Sheila. Review: Ontologies Application in Organizational Learning: A Literature Review. **Expert System with Applications: An International Journal**, Tarrytown, NY, USA, v. 39, n. 8, p. 7555–7561, 2012. DOI: 10.1016/j.eswa.2012.01.075.

VARMA, Vasudeva. Use of ontologies for organizational knowledge management and knowledge management systems. *In:* **ontologies**. [s.l.] : Springer, 2007. p. 21–47.

WAHID, S. Investigating Design Knowledge Reuse for Interface Development. *In:* PROCEEDINGS OF THE 6TH CONFERENCE ON DESIGNING INTERACTIVE SYSTEMS 2006, New York, NY, USA. **Anais** [...]. New York, NY, USA: ACM, 2006. p. 354–356. DOI: 10.1145/1142405.1142462.

WAHID, S.; SMITH, J. L.; BERRY, B.; CHEWAR, C. M.; MCCRICKARD, D. S.

Visualization of design knowledge component relationships to facilitate reuse. *In*: PROCEEDINGS OF THE 2004 IEEE INTERNATIONAL CONFERENCE ON INFORMATION REUSE AND INTEGRATION, 2004. IRI 2004. 2004, **Anais** [...]. [s.l.: s.n.] p. 414–419. DOI: 10.1109/IRI.2004.1431496.

WIERINGA, Roel; MAIDEN, Neil; MEAD, Nancy; ROLLAND, Colette. Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion. **Requir. Eng.**, Secaucus, NJ, USA, v. 11, n. 1, p. 102–107, 2005. DOI: 10.1007/s00766-005-0021-6.

WILSON, Paul; BORRAS, John. Lessons learnt from an HCI repository. **International Journal of Industrial Ergonomics**, [S. l.], v. 22, n. 4, p. 389–396, 1998. DOI: [https://doi.org/10.1016/S0169-8141\(97\)00093-0](https://doi.org/10.1016/S0169-8141(97)00093-0).

WOHLIN, Claes; RUNESON, Per; HÖST, Martin; OHLSSON, Magnus C.; REGNELL, Björn; WESSLÉN, Anders. **Experimentation in software engineering**. [s.l.] : Springer, 2012.

Appendix A

Artifacts used in KTID Evaluation Study

This appendix presents the artifacts we used in the evaluation study of KTID.

A.1 Instructions Document

	<p>UFES (Universidade Federal do Espírito Santo) NEMO (Núcleo de Estudos em Modelagem Conceitual e Ontologias)</p> <p><i>Evaluation study of the use of a computational tool to support knowledge management aspects in HCI design.</i></p>	
---	---	---

Student: Murillo Vasconcelos Henriques Bittencourt Castro

Advisors: Monalessa Perini Barcellos (UFES) e Ricardo de Almeida Falbo (UFES) (*in memoriam*)

I. Introduction

Human-computer interaction (HCI) design is focused on designing an interactive software system to support its user to achieve their goals through the interaction with the system. It involves knowledge from multiple fields, such as ergonomics, cognitive science, human factors, among others. Due to this diverse knowledge field, HCI design teams are frequently multidisciplinary, gathering people with different backgrounds and experiences, with their own technical terms and knowledge. Hence, they may have different conceptualizations about HCI design, which can hamper communication and knowledge transfer. Moreover, HCI design employs a huge amount of tacit knowledge, a type of knowledge that cannot be easily articulated and described for who holds it, aggravating to the difficulties in communication and knowledge transfer.

For instance, a designer may not be able to describe the reasons why he/she made certain design choice, or maybe he/she describe it in such a way that other designers, developers and project managers cannot correctly understand it. Therefore, the design may be not correctly implemented, and the knowledge employed to perform that task may be inaccessible for reuse in the future. In this context, Knowledge Management (KM) principles, practices, methods and tools may be useful, providing support to capture and represent knowledge in an accessible and reusable way, promoting knowledge from individual level to the organizational level.

In the context of the research in which this study is being carried, we proposed a **computational tool to support knowledge management aspects in HCI design**. The tool aims to allow designers to describe HCI design choices in HCI design solutions in a structured way, making this knowledge explicit and accessible. Next, we present the main functionalities of the tool.

Manage Design Objects: this use case aims to record information about design objects that are being developed and further associate them with their user requirements and design specifications. Figure 1 presents a screen which showing the list of design objects recorded in KTID and Figure 2 presents a screen in which we can record a new design object.

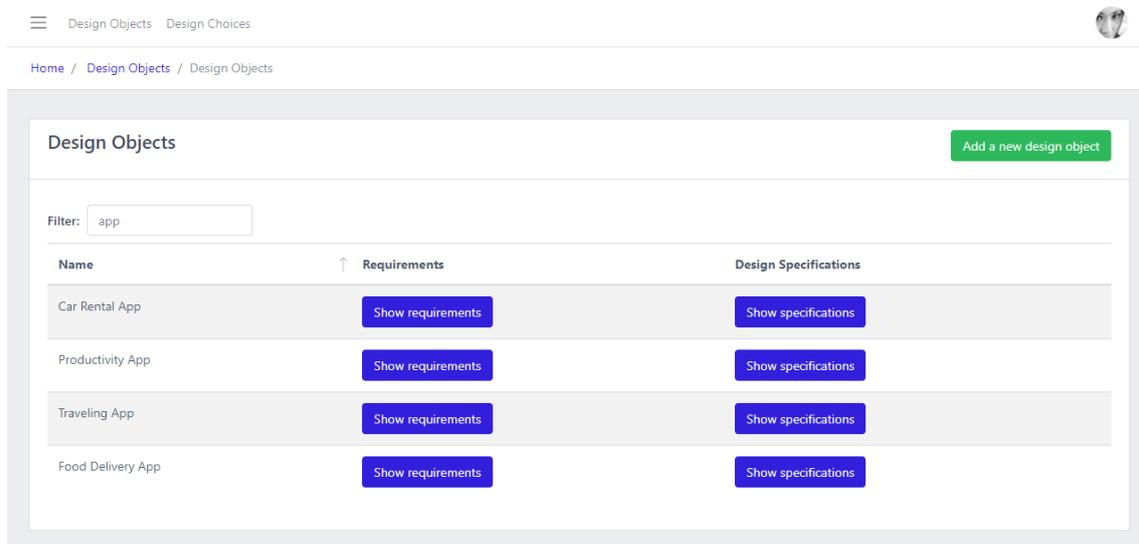


Figure 1 – List of design objects recorded in KTID.

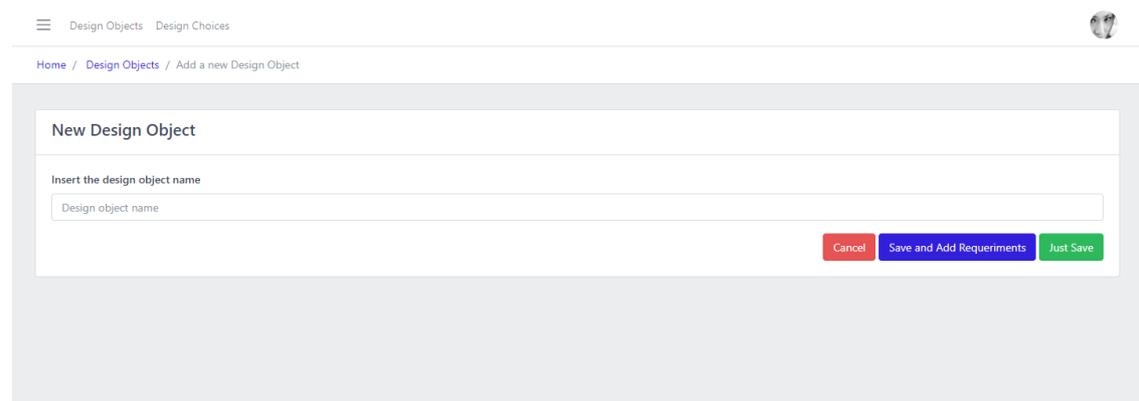


Figure 2 – Recording a new design object in KTID.

Manage User Requirements: this use case aims to record information about user requirements associated to a design object. Figure 3 presents a KTID screen which shows the list of user requirements for a specific design object and Figure 4 presents a screen in which we can record new user requirements.

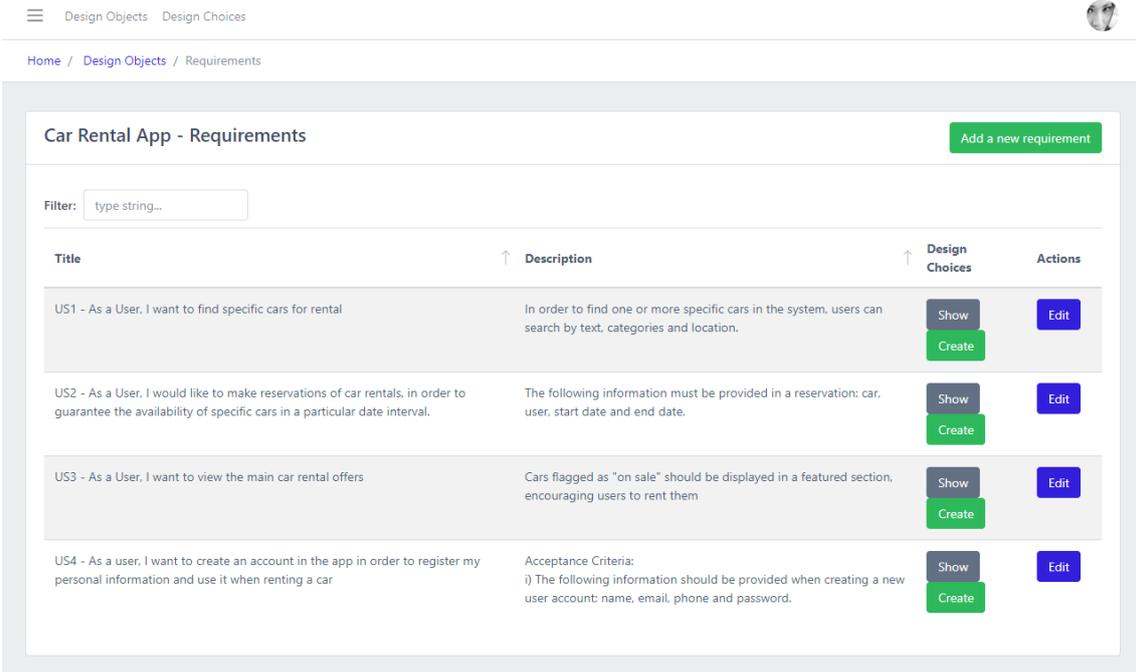


Figure 3 – List of user requirements for a design object recorded in KTID.

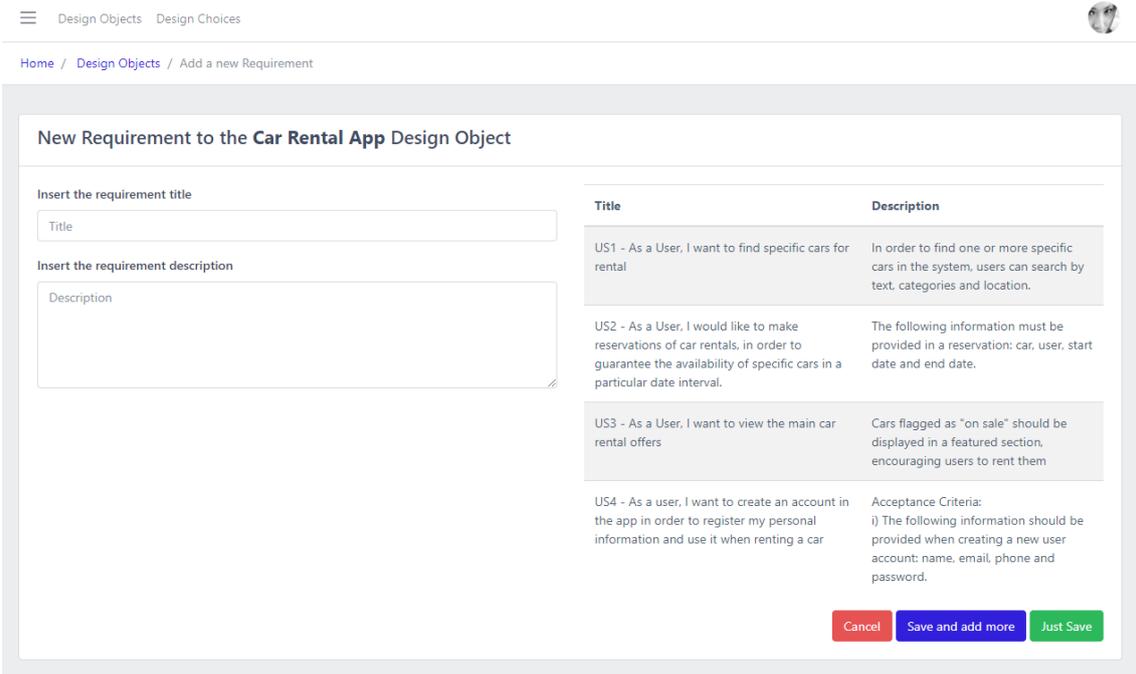


Figure 4 – Recording a new user requirement to a design object in KTID.

Manage Design Specifications: this use case aims to record design specifications, which consist of a set of image files. We can insert several specifications for the same design object, with different levels of fidelity (e.g., mockups, wireframes or sketches), viewport sizes (e.g., mobile phone, tablets or desktops) and version numbers. Figure 5 presents a KTID screen showing the list of files of a design specification as well its design choices and Figure 6 presents the screen in which a new design specification can be recorded.

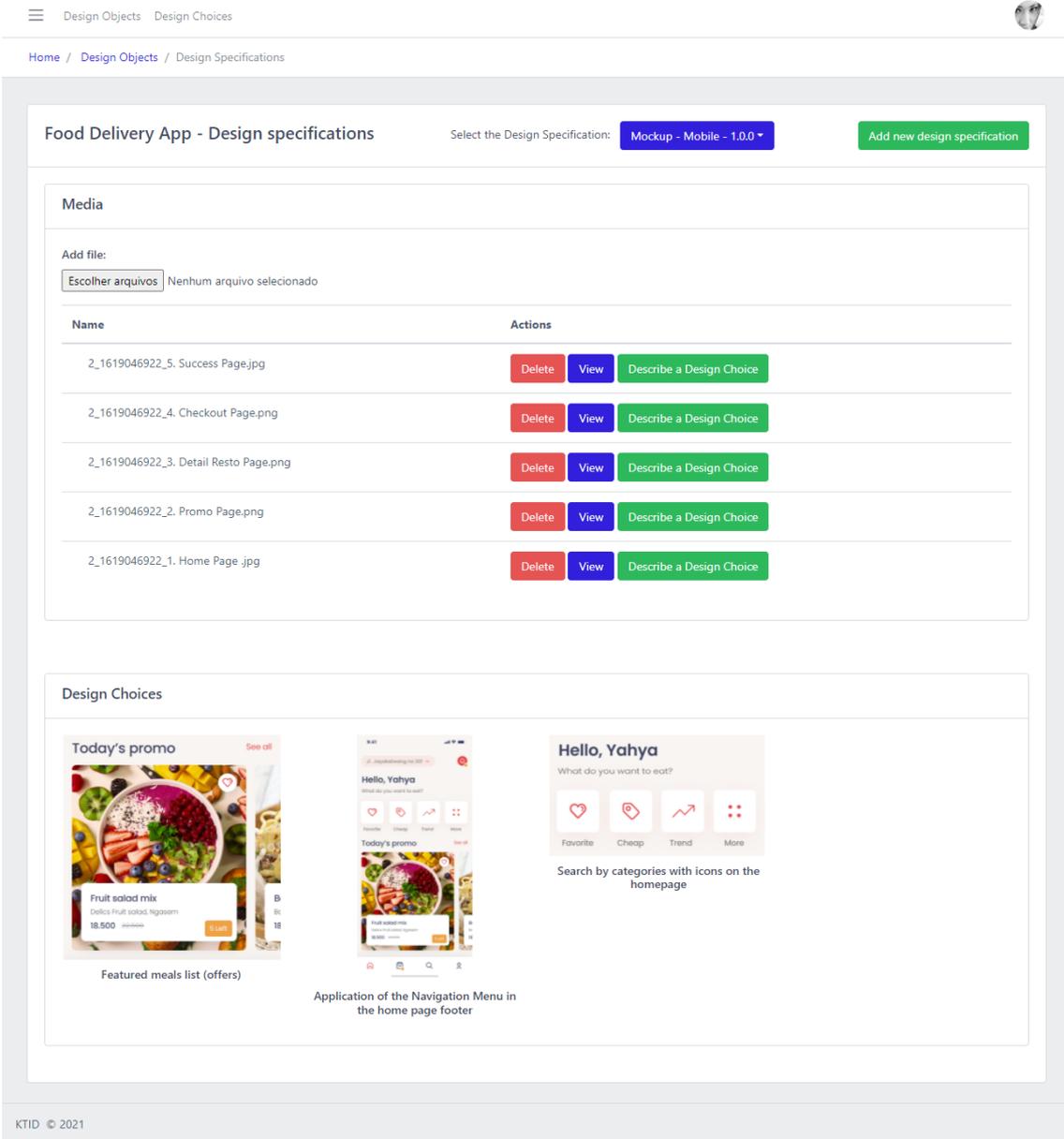


Figure 5 – Details of a design specification recorded in KTID.

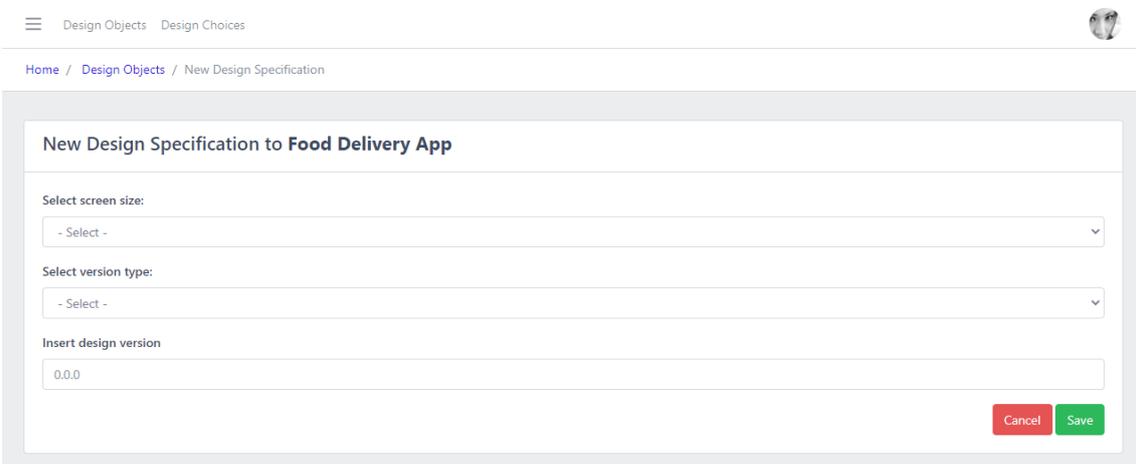


Figure 6 – Recording a new design specification for a design object in KTID.

Describe Design Choices: this is one of the main functionalities of KTID. It aims at allowing designers to identify, represent and describe design choices and related design components encoded in design specifications, as well link them with their associated user requirements. Figure 7 presents a KTID screen in which a design choice can be described and recorded for a design specification and Figure 8 presents a screen showing the details of a design choice recorded in KTID.

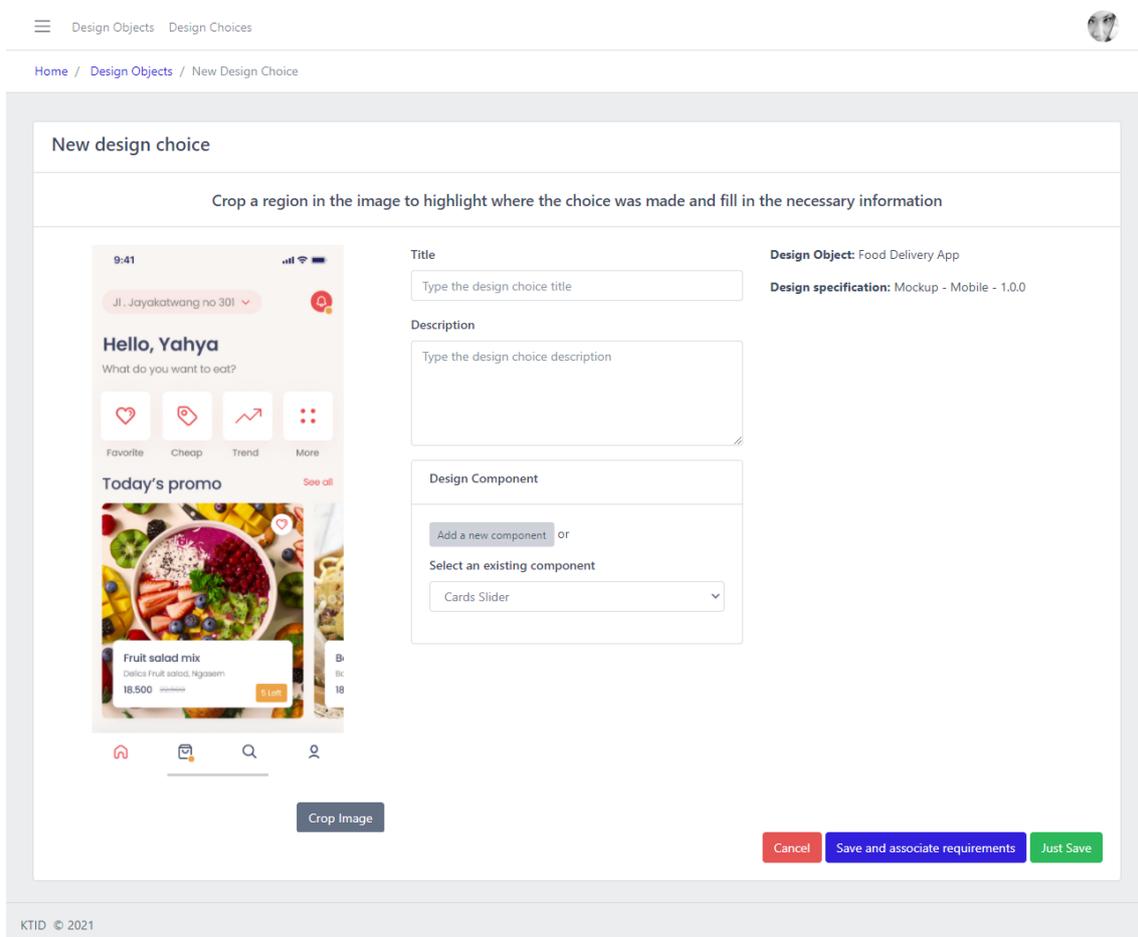


Figure 7 – Describing a design choice in KTID.

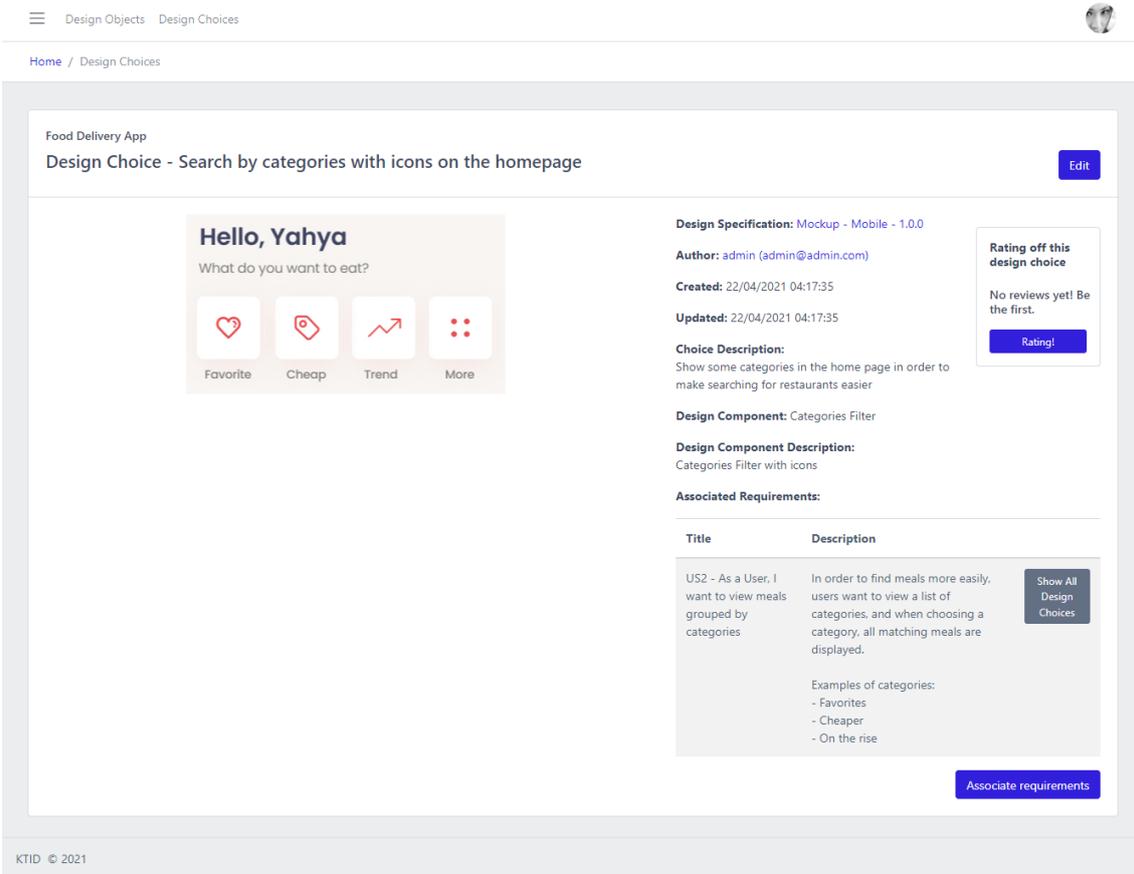


Figure 8 – Details of a design choice recorded in KTID.

Search for Design Choices: this is another important functionality of KTID. Figure 9 presents a KTID screen in which we can search for all design choices recorded in KTID. In this screen, we can visualize contextual details (e.g., related components and user requirements, the referred design object and design specification, author, date and time of creation and last update) about each choice. Each column of the table can be filtered or sorted, making it easier to find previous design choices related to contextual information similar to that of another specific context.

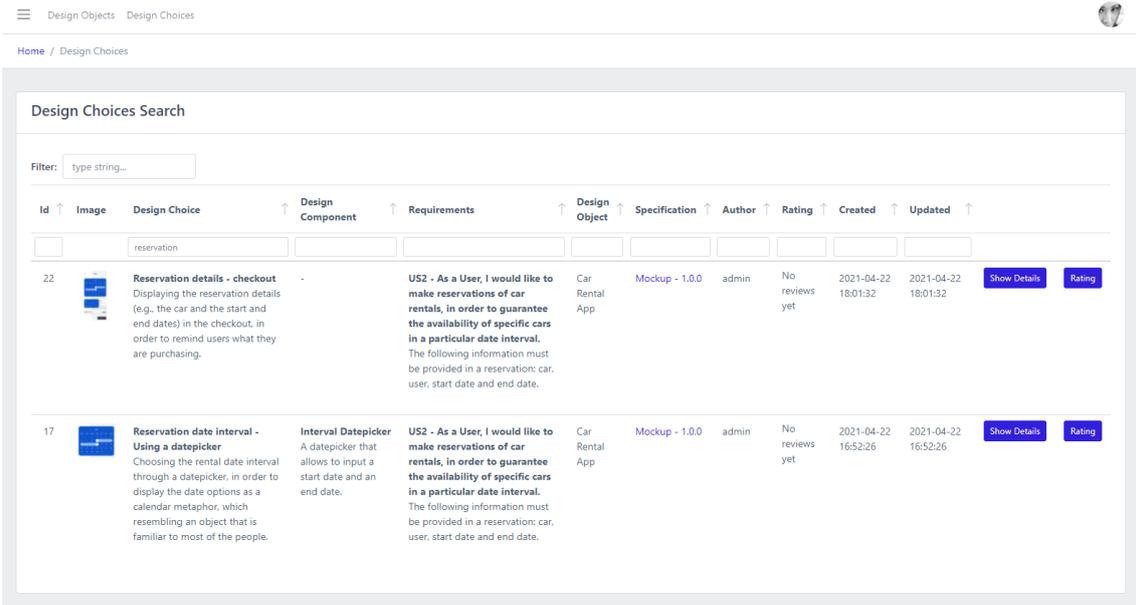


Figure 9 – Searching for design choices in KTID.

Evaluate Design Choices: once a designer finds a design choice satisfying his/her search criteria, he/she can evaluate the design choice rating how much it was useful in another context. By doing so, other designers can make decisions about using or not certain design choice based on its average rating. Figure 10 presents a KTID screen in which we can evaluate a design choice and figure 11 presents a screen showing the list of evaluations of a design choice.

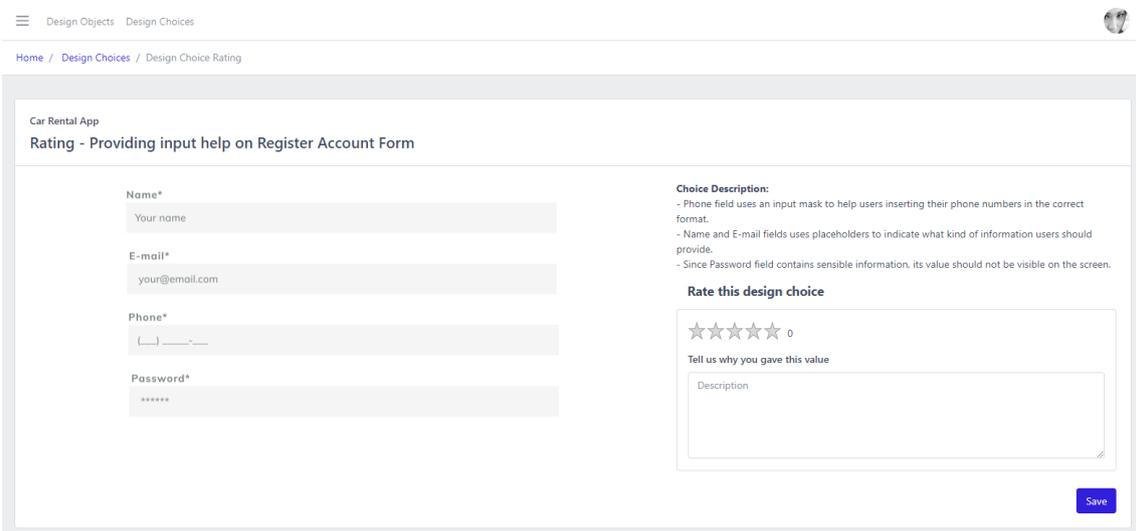


Figure 10 – Evaluating a design choice.

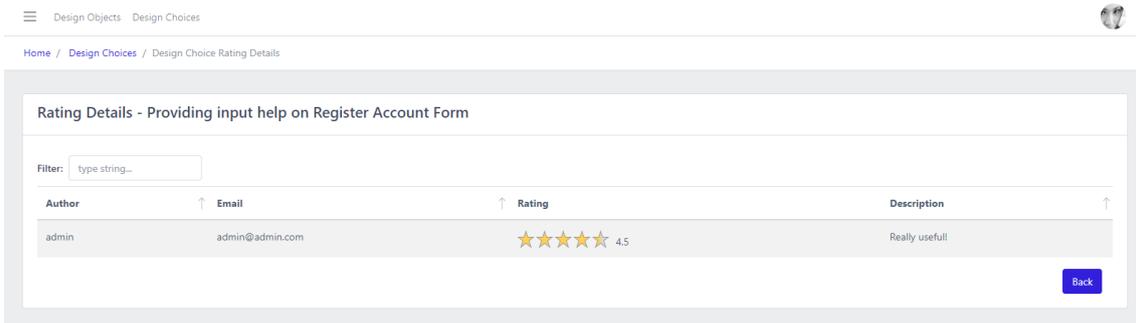


Figure 11 – Previous evaluations of a design choice.

II. Instructions

This study goal is to **evaluate the use of a computational tool to aid in HCI design**, considering its feasibility and utility in supporting knowledge management aspects. Hence, **this study is not concerned with evaluating the tool design itself**, but rather its use to aid HCI design providing support to knowledge representation, storage, retrieval and evaluation.

In order to carry out the study, the participant must use the tool as a support resource to perform a task that consists of the elaboration of a design solution for an interactive system. The tool is available through the link <http://bit.ly/KTID-tool>. To access it, you must log in using the user “**admin@admin.com**” and the password “**password**”. After completing the task, the participant must complete the questionnaire available at <http://bit.ly/KTID-Evaluation>.

Below we present a description of the domain for which the interactive system is being developed, as well as detailed instructions for carrying out the task.

Domain description: A company that offers movie streaming services wants to build a platform (*FilmFlix*) where its customers can access the company’s catalog and watch the available movies. The platform contains the following information for each film in the catalog: name, release date, cover image, synopsis and genre. From its customers, the company wants to know their name, date of birth, e-mail and profile picture. When accessing the catalog, customers can book movies that they have interest in watching in the future. Such films are available in a list of movies called “Next films”. In addition, customers can consult the catalog to search for specific films, through textual search or by genre selection. The platform should also provide customers with a list of the most recent releases and a featured list with the most popular films. When the customer chooses to watch a certain film, the platform must open the video player and play it.

Instructions for carrying out the task: Considering the domain presented above, the participant must develop a design solution in the form of **wireframes for the graphical interface of the initial screen of the *FilmFlix* platform**. Wireframes are artifacts that outline the basic structure of the graphical interface of an interactive system (e.g., how the elements are visually organized when viewed on the screen), represented in the form of low-fidelity sketches, that do not address specific details such as colors and typography. Figure 12 shows an example of a wireframe for the *Facebook* profile page.

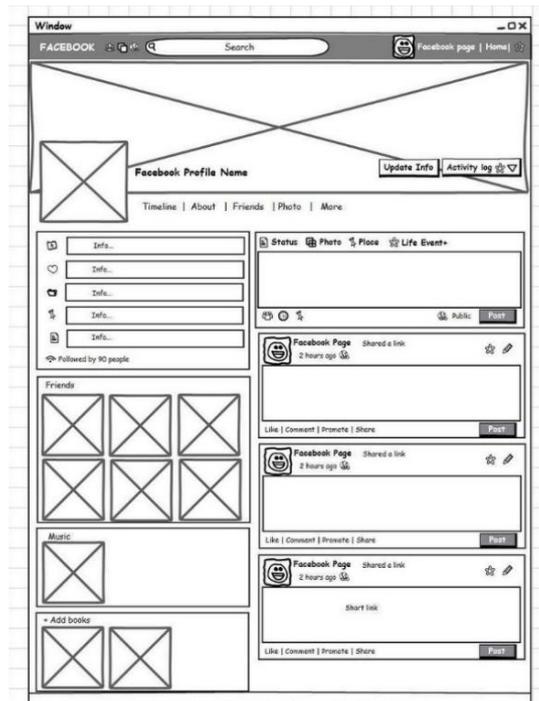


Figure 12 – Wireframe of the *Facebook* profile page⁸.

Since the platform will be accessed through both mobile devices and desktop computers, the design solution to be produced by the participant must contemplate specific aspects of the interaction with each type of device (e.g., changing the layout of the elements due to the difference in viewport size or using other elements that are more suitable for touch interaction on mobile devices). The steps that must be followed to perform the task are described below. The order in which they are listed is only suggestive and should not necessarily be followed. In addition, the participant can repeat a step more than once if deemed necessary.

- Access the user requirements page at <http://bit.ly/FilmFlixRequirements> and verify the specified user requirements for *FilmFlix* based on its domain description.

⁸Adapted from: <https://www.flickr.com/photos/mockupbuilder/8705902051> (accessed on April 22, 2021)

- Access the design choices search page at <http://bit.ly/DesignChoicesSearch> and verify which design choices available in the tool can be useful to produce *FilmFlix* design solution.
- Produce the proposed design solution for *FilmFlix*. We recommend using a computational tool that allows to export the solution as an image file. If the participant prefers, the solution can be drawn on paper, since the drawing is digitalized or photographed.
- Upload the produced solution in the *FilmFlix* design specification.
- Describe the design choices made by the participant during the elaboration of the design solution and associate them with the related user requirements.
- Evaluate previous design choices that were reused or adapted in the solution produced by the participant.
- Answer the feedback questionnaire available at <http://bit.ly/KTID-Evaluation>.

A.2 Consent Form

Evaluation Study

Evaluation study of the use of a computational tool to support knowledge management aspects in HCI design.

Consent Form

This study is being carried out in the context of the master's research of student Murillo Vasconcelos Henriques Bittencourt Castro, carried out in the Graduate Program in Informatics at the Federal University of Espírito Santo (UFES), advised by the professors Monalessa Perini Barcellos and Ricardo de Almeida Falbo (in memoriam). The goal of the study is to evaluate the use of a computational tool to support knowledge management aspects in HCI design, in the context of interactive software systems development.

After accepting this consent form, some questions will be raised. The confidentiality of individual data provided in the study is guaranteed. The data will be used to carry out the research and will be not used as a personal or professional assessment. Participants' name (optional) and e-mail are requested only for contact purposes, in case there is a need to clarify any of the responses.

Despite being invited, participation in this study is voluntary, being entitled not to want to participate or abandon the study at any time.

By clicking on "Next", I declare that I am over 18 and voluntarily participate in this study, having read the information contained in this term before participating.

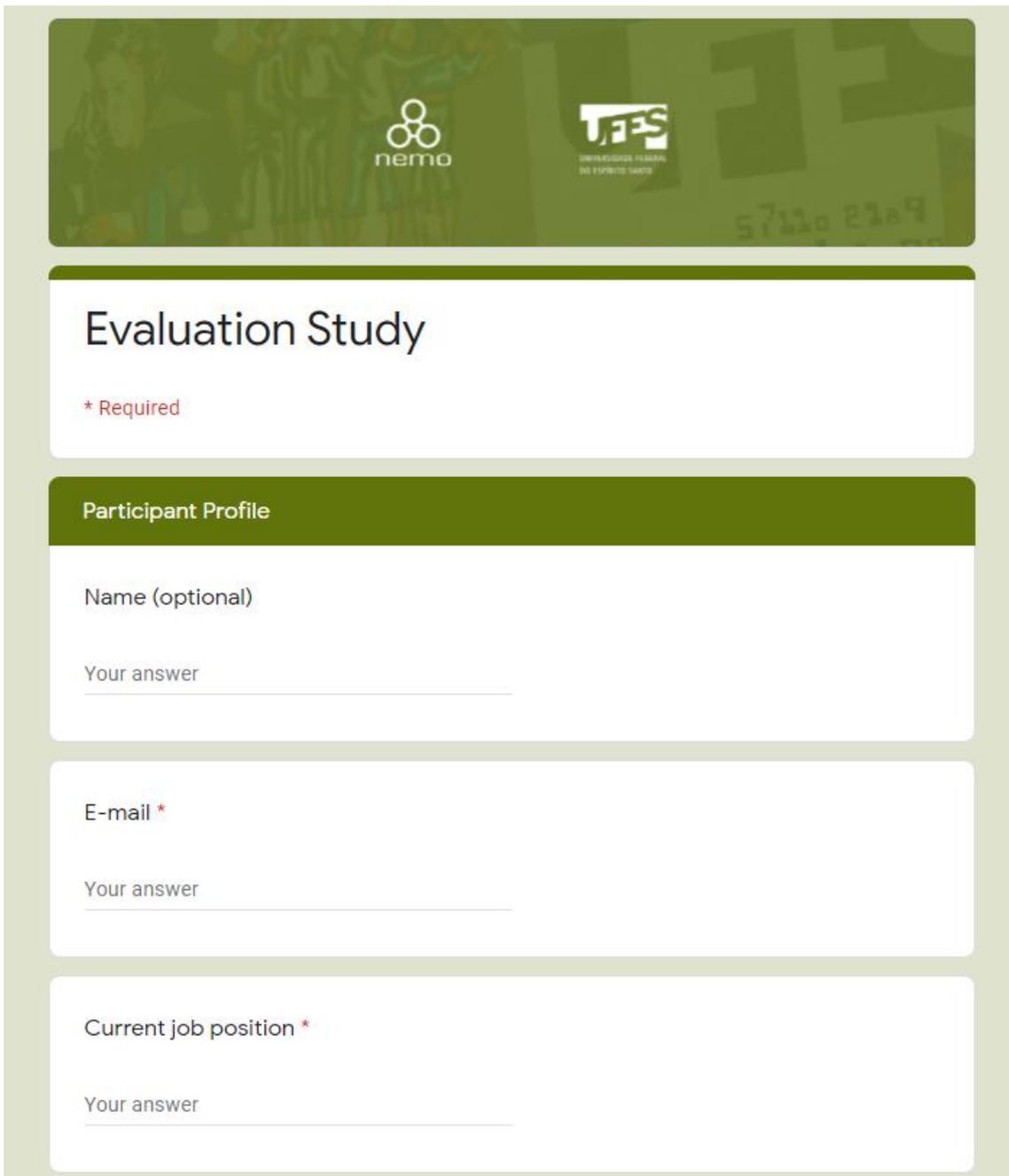
[Next](#) Page 1 of 3

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

Google Forms

A.3 Participants Profile Form



The form is titled "Evaluation Study" and includes a header with logos for "nemo" and "UFES" (Universidade Federal do Espírito Santo). The main content is organized into sections: a title section, a "Participant Profile" section, and three input fields for "Name (optional)", "E-mail", and "Current job position". Each input field is preceded by the text "Your answer" and has a horizontal line for text entry.

Evaluation Study

* Required

Participant Profile

Name (optional)

Your answer _____

E-mail *

Your answer _____

Current job position *

Your answer _____

Education

Academic degree *

Choose your highest academic degree among the options below

- Elementary School
- High School
- Bachelor's degree
- Specialization
- Masters' degree
- Ph.D. degree

Situation of the highest academic degree *

- Complete
- Incomplete

Course or study area of the highest degree of the indicated academic education

Your answer _____

Theoretical knowledge and practical experience in HCI design

HCI design theoretical knowledge *

Choose the option which is most adequate to your knowledge about HCI design

- None
- Low (i.e., knowledge acquired by yourself through books, videos or other materials)
- Medium (i.e., acquired mainly during courses or undergraduate research)
- High (i.e., you are an expert or have a certification, Masters or Ph.D. degree related to HCI design)

Practical experience in HCI design *

In this question, consider how long you have been practicing HCI design activities

- None
- Low (less than 1 year)
- Medium (between 1 year and 3 years)
- High (more than 3 years)

Theoretical knowledge and practical experience with knowledge management

Theoretical knowledge about Knowledge Management *

Choose the option which is most adequate to your knowledge about Knowledge Management

- None
- Low (i.e., knowledge acquired by yourself through books, videos or other materials)
- Medium (i.e., acquired mainly during courses or undergraduate research)
- High (i.e., you are an expert or have a certification, Masters or Ph.D. degree related to HCI design)

Have you ever used knowledge management resources (e.g., a wiki, a design solutions repository, guidelines, examples repositories, or a knowledge management system) to support your HCI design practice *

- No
- Yes, on my own initiative, there were no policies, best practices or principles defined by the organization
- Yes, following institutionalized organizational practices (i.e., the organization had defined policies, best practices or principles to the usage of such resources)

A.4 Feedback Questionnaire

Feedback Questionnaire

1. Did you find out in the tool relevant knowledge to produce the proposed design solution? *

Yes

No

*

Justification:

Your answer

2. Did you include in the tool knowledge about the solution produced by you that may be relevant to produce possible future solutions? *

Yes

No

*

Justification:

Your answer

3. Do you consider the tool adequate for supporting the following KM activities? *

	Yes	Partially	No
KNOWLEDGE REPRESENTATION (allows making explicit useful knowledge for HCI design)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
KNOWLEDGE STORAGE (allows storing useful knowledge for HCI design)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
KNOWLEDGE RETRIEVAL (allows to retrieve and access stored knowledge)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
KNOWLEDGE EVALUATION (allows to assess whether the knowledge accessed is useful)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

*

Justification:

Your answer

4. What was your perception about the tool's utility? *

- Very useful
- Useful
- Neutral
- Useless
- Very useless

*

Justification:

Your answer

5. Do you consider that the use of the tool to carry out the proposed task contributed to any of the following results? *

	Yes	No
GREATER QUALITY of the produced solution	<input type="radio"/>	<input type="radio"/>
LESS EFFORT employed to produce the solution	<input type="radio"/>	<input type="radio"/>
LESS TIME spent to perform the task.	<input type="radio"/>	<input type="radio"/>

*

Justification:

Your answer

6. What was your perception of the tool's ease of use? *

- Very easy
- Easy
- Neutral
- Hard
- Very hard

*

Justification:

Your answer _____

7. What was your perception regarding the feasibility of using the tool? *

- Very feasible
- Feasible
- Neutral
- Infeasible
- Very infeasible

*

Justification:

Your answer _____

8. Would you recommend the use of the tool to other people? *

Yes

No

*

Justification:

Your answer

9. Based on your experience using the tool, what BENEFITS do you believe it would provide if used in the organization for which you work? *

Your answer

10. Based on your experience using the tool, what DIFFICULTIES do you believe it would provide if used in the organization for which you work? *

Your answer

11. Which SUGGESTIONS do you have for the improvement / evolution of the tool in order to improve the support for knowledge management aspects in assisting the HCI design? *

Your answer
