Using Ontologies to aid Knowledge Sharing in HCI Design

Murillo V. H. B. Castro, Monalessa P. Barcellos, Ricardo de A. Falbo
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department, UFES
Vitória, Brazil
murillo.castro@aluno.ufes.br,{monalessa,falbo}@inf.ufes.br

Simone D. Costa
Computer Science Department, UFES
Alegre, Brazil
simone.costa@ufes.br

ABSTRACT

Developing interactive systems is a challenging task that involves concerns related to the human-computer interaction (HCI), such as usability and user experience. Therefore, HCI design is a core issue when developing such systems. It often involves people with different backgrounds (e.g., Arts, Software Engineering, Design), which makes knowledge transfer a challenging issue. Ontologies have been acknowledged as a successful approach to represent domain knowledge and support knowledge-based solutions. Hence, in this work, we propose to explore ontologies to represent structured knowledge and improve knowledge sharing in HCI design. We briefly present the Human-Computer Interaction Design Ontology (HCIDO), a reference ontology that addresses HCI design aspects that connect HCI and Software Engineering concerns. By making knowledge related to the HCI design domain explicit and structured, HCIDO has helped us to develop KTID, a tool that aims to support capturing and sharing useful knowledge to aid in HCI design. Preliminary results indicate that the tool may be particularly useful for novice HCI designers.

CCS CONCEPTS

• **Human-centered computing → HCI theory, concepts and models**; *Interactive systems and tools.*

KEYWORDS

HCI Design, Ontology, Knowledge

ACM Reference Format:
Murillo V. H. B. Castro, Monalessa P. Barcellos, Ricardo de A. Falbo and Simone D. Costa. 2021. Using Ontologies to aid Knowledge Sharing in HCI Design. In *XX Brazilian Symposium on Human Factors in Computing Systems (IHC'21), October 18–22, 2021, Virtual Event, Brazil.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3472301.3484327

1 INTRODUCTION

Designing quality interactive computer systems is a challenging task, which involves addressing HCI aspects (e.g., usability, user experience, communicability, accessibility, among others) to support users in achieving their goals through the interaction with the system [5]. HCI design comprises practices, principles and knowledge from multiple fields, such as ergonomics, cognitive science, sociology, human factors and computer science [31]. Due to the diverse body of knowledge involved in HCI design, interactive system development teams are often multidisciplinary, joining people from different backgrounds, with their own technical language, terminology and knowledge. As a consequence, even the understanding of the product may be conflicting among different stakeholders, which hampers communication and knowledge transfer [5, 22].

Although HCI design and Software Engineering (SE) have different concerns on the development of interactive systems (HCI design is more user-centered while SE is more system-centered), there is a strong connection between these areas [17]. Hence, it is important to reach a consensual understanding on the meaning of terms related to both areas to avoid semantic conflicts and facilitate knowledge sharing. For example, an HCI designer may refer to the user interface as what is seen through the graphical elements displayed on the screen, while a developer may refer to the user interface as the portion of the code that produces the graphical elements displayed in the screen.

Knowledge Management (KM) principles and practices can be helpful to address knowledge-related and communication challenges in HCI design. There are some initiatives of using KM to enable knowledge replicability and improve communication in the HCI design context and, in most of them, KM has been used with the ultimate purpose of improving software quality and design process efficiency. However, KM solutions have been narrowly explored in HCI design and have faced difficulties mainly related to the lack of consensus on the understanding of HCI design aspects [6].

The use of ontologies contributes to capture and organize knowledge to deal with knowledge-related and communication problems. In the HCI context, they have been applied to aid knowledge representation in some sub-domains (e.g., user interface), support interface adaptation, aid in interaction design and evaluation, among others [9]. However, there is still a need of properly understanding HCI design and its relation with SE, so that designers and developers can agree on the same conceptualization of the interactive system under development and, thus, better communicate and exchange knowledge with each other.

In view of the above, we advocate that ontologies are a promising approach to aid in HCI design. Thus, we have developed the Human-Computer Interaction Design Ontology (HCIDO), a reference ontology that addresses HCI design mental (i.e., what is in the designer's mind) and materialized (i.e., the artifacts produced based on what is in the designer's mind) aspects and connects them to SE concepts, providing an integrated view of HCI design and SE

IHC'21, October 18–22, 2021, Virtual Event, Brazil

ACM ISBN 978-1-4503-8617-3/21/10…$15.00
https://doi.org/10.1145/3472301.3484327

related aspects. Moreover, considering KM principles that seek to transform individual and implicit knowledge into organizational and explicit knowledge, we have used HCIDO to develop KTID (Knowledge Supporting Tool for Human-Computer Interaction Design), a computational tool that provides support to represent and share HCI design knowledge among stakeholders, such as designers, developers and project managers. HCIDO has provided domain knowledge and facilitated KTID development. KTID was used by two designers and the preliminary results indicate that the tool may be particularly useful for novice designers, who need to learn from previous experiences to create new designs. On the other side, experienced designers may prefer a more creative and individual process, being more willing to share knowledge than to reuse knowledge recorded in the tool.

The remainder of this paper is organized as follows: Section 2 provides the background for the paper; Section 3 discusses some related works, Section 4 briefly presents HCIDO; Section 5 introduces KTID; and Section 6 concludes the paper.

2 BACKGROUND

2.1 HCI Design and Knowledge Management

HCI can be defined as the discipline responsible for the analysis, design, implementation and evaluation of interactive computer systems for human use [22]. An interactive computer system (also referred here as "interactive system") is a combination of hardware and software that receives input from and communicates output to users [15]. According to [10], the communication between the user and the interactive computer system is the interaction itself. User and system are, thus, participants in the interaction.

HCI design focuses on how to design interactive computer systems to support users to achieve their goals through the interaction between them and the system [31]. 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 [15].

HCI design can 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 necessary design for users to achieve their goals. Schneider (2009) [25] defines knowledge as a human specialty stored in people's minds, acquired through experience and interaction with their environment. According to [20], knowledge can be classified in two types: tacit and explicit. *Tacit knowledge* represents the subjective and non-documented knowledge that lies in people's mind, which 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.

Many times, organization's knowledge is undocumented, being represented through the skills, experience and knowledge of its professionals, which makes its use and access limited and difficult [18, 23]. *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 organization [18, 23, 25].

2.2 Ontologies

An *ontology* is a formal and explicit specification of a shared conceptualization [28]. 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 [27].

An important distinction differentiates ontologies as conceptual models, called *reference ontologies*, from ontologies as computational artifacts, called *operational ontologies* [14]. A reference ontology is constructed with the goal of making the best possible description of the domain in reality, 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 large and complex domains, ontologies can be organized in an ontology network (ON), which consists of a set of ontologies connected to each other through relationships in such a way to provide a comprehensive and consistent conceptualization [30]. The ontology presented in this paper (HCIDO) reuses concepts from (and, thus, is connected to) two ONs: the Human-Computer Interaction Ontology Network (HCI-ON) [9], which contains ontologies addressing HCI sub-domains (e.g., HCI phenomenon, HCI Evaluation), and the Software Engineering Ontology Network (SEON) [24], which includes ontologies addressing ES sub-domains (e.g., Software Requirement, Software Process, System and Software).

3 RELATED WORKS

In the literature, there are some works proposing ontologies related to HCI design. However, different from HCIDO, they do not focus on describing the HCI design itself or representing its relation with SE. In [26], for example, an ontology that addresses interaction aspects is proposed to describe interactive behaviors on user interfaces, aiming to support test automation of interactive systems functional requirements. In [19], in turn, *UI*^{2}*Ont* addresses concepts related to user interface (such as components and activities) and defines taxonomies (i.e., classifications) to those concepts. Other ontologies, although addressing HCI design itself, focus on HCI design in specific contexts, such as design of web applications [1, 2], haptic devices [16] and gesture-based interactions [8].

We did not find any work using ontologies to aid KM in HCI design. The work more closely related to ours is [29], which proposes a process to automate model transformation of a task description into an interaction description, based on metamodels derived from ontologies that represent knowledge about the user interface design

process. In that work, knowledge was identified, uniformized and captured through a KM strategy in which ontologies are used as a mean to support model transformation. Different from our work, [29] is not concerned with knowledge sharing. In our work, the ontology (HCIDO) is used to provide HCI design domain knowledge that contributed to a better understanding of the domain and to develop a tool to support knowledge representation and sharing.

4 THE HUMAN-COMPUTER INTERACTION DESIGN ONTOLOGY (HCIDO)

HCIDO provides a well-founded consensual conceptualization of HCI design, addressing both mental and corresponding materialized aspects. In its current version, HCIDO focuses on the design of software aspects of the user interface of interactive systems. Hence, the design of hardware aspects, the interaction interaction itself and user experience facets have not been covered by HCIDO yet. HCIDO conceptualization was developed based mainly on works by Ralph and Wand [21] and Guarino [12], which regard design core concerns, as well as on ISO standards (e.g., [15]), which are particularly concerned with HCI. It addresses a knowledge intersection between SE and HCI design by reusing SE concepts from SEON [24] and HCI core concepts from HCI-ON [9]. On one hand, reusing concepts from existing ontologies helped accelerate HCIDO development because, by reusing SE and HCI existing knowledge, we were able to focus on HCI design itself. On the other hand, HCIDO is integrated to HCI-ON [9], contributing to grow the network conceptualization and to connect it to the SE conceptualization provided by SEON [24]. Being a well-founded reference ontology, HCIDO is grounded in the Unified Foundational Ontology (UFO) [13]. Discussion regarding the grounding of HCIDO in UFO falls outside the scope of this paper and can be found in [7].

HCIDO was developed by following SABiO [11]. In order to establish the ontology scope, we defined competency questions, which are questions the ontology is intended to answer. Some examples of competency questions defined to HCIDO are: How does an HCI designer reason about the object being designed? What is an HCI design specification? Which are the components of an HCI design object? What is described in an HCI design specification? What is the motivation for an HCI design choice? How can an HCI design object be implemented from an HCI design specification?

A fragment of HCIDO conceptual model is presented in Figure 1. Due to space limitation, the figure does not include concepts related to mental aspects and presents only HCIDO concepts relevant to this paper. Moreover, the figure only shows SEON and HCI-ON concepts directly related to the included HCIDO concepts. In the figure, HCIDO concepts are presented in orange, HCI-ON concepts in yellow and SEON concepts in blue. In the model description, concepts from HCIDO are written in **bold**, while concepts from SEON are written in *italics* and from HCI-ON in *underline italics*.

To better understand HCIDO concepts, it is necessary, first, to understand some SE-related concepts from SEON [24] (the ones in blue in Figure 1). A *Person Stakeholder* is a person interested or affected by software development activities or their results. *Software Artifacts* are objects intentionally produced to serve a given purpose in the context of a software project or organization. They can be classified according to their nature. A *Software Item* is a piece of software, produced during the software process, not necessarily a complete product (i.e., it can be an intermediary result such as a system 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 *Software System* (e.g., a system to buy airline tickets) is a subtype of *Software Item* that is constituted of *Programs*. A *Program*, in turn, is a *Software Item* not considered a complete *Software System* (e.g., the system component to select available flights in a given date), which aims at producing a certain result, in a particular way, through its execution on a computer.

A *Requirement* is a goal to be achieved, representing a capacity needed for the users (e.g., buy airline tickets). When a *Requirement* is recorded in some kind of *Software Artifact*, there is a *Requirement Artifact* describing that *Requirement*. A *Requirement Artifact* is an *Information Item* responsible for keeping relevant information for human use (e.g., a sentence stating that "the system must allow the user to buy selected airline tickets").

In addition to concepts related to SE, it is also necessary to understand some HCI core concepts [9]. An *Interactive Software System* is a *Software System* constituted (among others) of *User Interface Programs*, which are *Programs* that handle the *User Interface*. The *User Interface* comprises all parts of the computer system that users have contact with, physically, perceptually or conceptually [3]. Users interact with *Interactive Software Systems* to achieve *User Goals*, which represent needs intended to be satisfied by the system.

In the context of HCI design, the **HCI Designer** is a *Person Stakeholder* responsible for creating **HCI Design Specifications**, which are *Software Artifacts* describing how an **HCI Design Object** must be materialized (in terms of HCI aspects). In HCIDO, the **HCI Design Object** is an *Interactive Software System*, i.e., the object being designed is an interactive system. Thus, an **HCI Design Specification** describes how a particular *Interactive Software System* should be. An **HCI Design Specification** contains one or more **HCI Design Choices**.

HCI Design Choices are *Information Items* that describe particular choices made by the **HCI Designer** concerning 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 (e.g., the fragment of a sketch showing the fields of a form arranged in two columns; a sentence written in a document describing the expected behavior after a form submission). Three subtypes of **HCI Design Specifications** are defined in HCIDO: **Wireframes**, **Mockups** and **Functional Prototypes**. As shown in Figure 1, this is an incomplete generalization set (indicated in the figure by {incomplete}), i.e., there are other types of **HCI Design Specifications** besides the ones represented in the conceptual model. 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

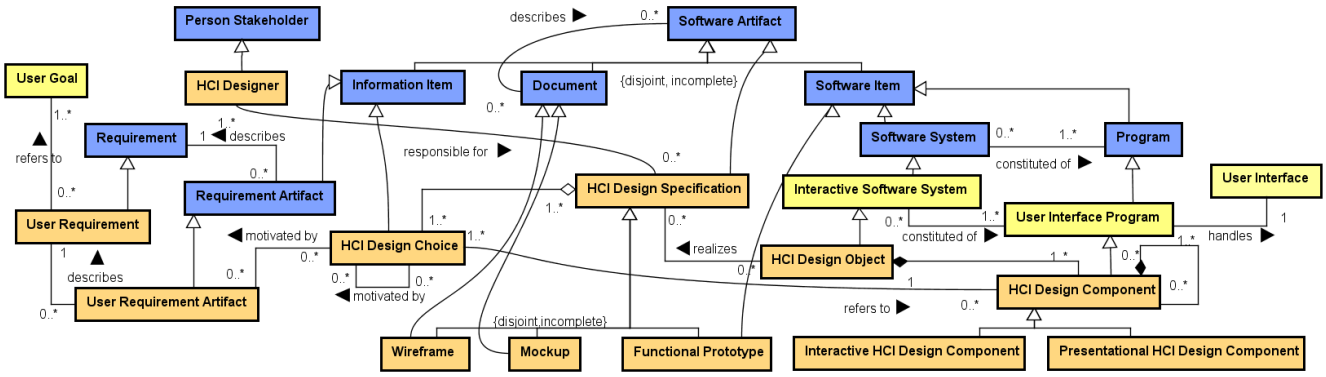

Figure 1: Fragment of HCIDO conceptual model.

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. 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.

HCI Design Choices can be motivated by previous **HCI Design Choices** (e.g., the selection of a certain set of colors to be used in a screen can motivate the use of the same set of colors in other screens) or by **User Requirements Artifacts** (e.g., user stories), which are *Requirement Artifacts* that describe **User Requirements**. For example, the decision of presenting a banner with new products at the top of a screen can be motivated by the user story stating that the user wants to be proactively informed about new products. Hence, **User Requirements**, are *Requirements* that refer to *User Goals*. It is important to highlight that the motivation for the **HCI Design Choices** is not always explicit in real-world situations (e.g., when design choices are motivated by designer's tacit knowledge).

An **HCI Design Object** is composed of **HCI Design Components**, which are *User Interface Programs* that implement elements that can be perceived or actioned by users through the *User Interface* and are referred on **HCI Design Choices**. 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 **Presentational HCI Design Component** (e.g., a text label) aims to present information that can be perceived through user's senses. An **Interactive HCI Design Component** (e.g., a button), in turn, is expected to be actioned (or not) in certain scenarios, according to the actions user perform during the interaction with 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.

As we explained before, although not shown in Figure 1, HCIDO also addresses concepts related to mental aspects involved in HCI design. For example, both the design object and its specification exist in the designer's mind before being materialized as the objects and artifacts showed in Figure 1. In fact, there may be situations in which the design choices and specifications are not materialized as artifacts (i.e., they exist only in the designer's mind).

By following SABiO [11], after developing HCIDO, to verify if the ontology properly covers the intended domain and is able to represent real-world situations, we performed verification and validation activities using assessment by human and data-driven approaches, as suggested in [4]. After that, we used HCIDO as a basis to develop a tool to help knowledge capture and sharing in HCI design. The tool is presented in the next section.

5 KTID: KNOWLEDGE SUPPORTING TOOL FOR HUMAN-COMPUTER INTERACTION DESIGN

As we discussed before, HCI design involves a lot of knowledge that may not be easily accessed as it lies in the designer's mind. Considering previous experiences of one of the authors working in a multidisciplinary team containing designers and developers and, by analyzing real-world situations in the light of the HCIDO conceptualization, we observed that sometimes it is not easy to identify all design choices encoded in a design specification because design specifications are often viewed as a whole and not as an aggregation of several individual choices. This makes it difficult to get knowledge about the decisions made until getting the design specification as a whole and, as a consequence, hampers the reuse of the knowledge behind these choices when creating other design objects. This motivated us to develop KTID, a tool that supports HCI designers to describe, share and retrieve knowledge related to choices made when designing HCI aspects of interactive systems.

HCIDO contributed to KTID development mainly by (i) providing the understanding of the tool application domain (i.e., HCI design); and (ii) serving as a basis to develop KTID conceptual model. Concerning (i), HCIDO conceptualization allowed us to spend less effort in KTID conception because the ontology provided knowledge about the domain of interest. As for (ii), by using HCIDO to develop KTID conceptual model, we were able to create a tool based on an HCI design general conceptualization instead of on a particular application context (e.g., HCI design in a specific organization). This way, KTID can be suitable for more diverse

HCI design situations. Moreover, we reused classes and relations from HCIDO conceptual model (making some adjustments, such as adding properties to the classes and creating a new class to record ratings), which demanded less effort than to create the conceptual model from scratch.

KTID was developed using a template theme[1] built over the frameworks Laravel[2] and Vue.js[3]. As main features, KTID allows HCI designers to record design specifications and design choices and inform the motivations (e.g., requirements or other choices) that led them to make such choices (e.g., the designer can record the chosen component (and related information) to be used, in order to meet a certain requirement in a design specification created for a particular interactive system). This feature aims to support knowledge capture and structuring so that it can be accessed and reused by others. Figure 2 illustrates the recording of a design choice in KTID regarding a "register account form" component, which was cropped from a mockup for a particular interactive system.

In the figure, the description of the choice aims to differentiate the meaning of the information displayed inside each field of the form. For example, the phone field contains an input mask, which should be visible while the user is typing, while name and email fields contain input placeholders, which provide examples of possible inputs to users and disappear as users start typing. Although these information have different semantics, they are syntactically represented in the same way in the mockup, relying on a shared and implicit understanding between who designed and who reads the mockup to make the distinction between their meanings. Hence, KTID aims at providing means to make this understanding explicit and shared between HCI design stakeholders.

When creating an HCI design, designers can also search for recorded design choices to reuse (or be inspired by) them. Designers can also evaluate the design choices by indicating, in a five-star scale, if they found them useful. These features aid in knowledge sharing. Figure 3 shows the KTID page used to search for design choices. Each column of the table can be filtered or sorted, making it easier for designers to find relevant information considering their needs. Developers can also use KTID to improve communication with designers. For example, a developer implementing a design choice can use the tool to retrieve design choice information to better understand its details and the motivations that led the designer to make it. Developers can also verify which design choices need to be implemented to satisfy a certain requirement that should be met.

We developed the current version of KTID aiming to reach a minimum set of features to allow us to verify if the tool can be viable and useful. Hence, this first version of the tool has some limitations that may affect its use, such as the lack of update and delete operations for some use cases. As a preliminary evaluation, KTID was used by two designers (one novice and one experienced). We provided the description of a particular interactive system and they were asked to create a wireframe to that system using KTID to record and reuse knowledge about design choices. The novice designer informed that the tool was useful and, since he reused knowledge recorded in the tool, he was motivated to record knowledge about the choices he made in the created wireframe. The experienced designer, in turn, said that he did not use knowledge available in the tool because he did not need it to create the proposed wireframe. He said he could record his knowledge to future use, but he believes that this may promote some kind of standardization and prevent other designers from being more creative. The designers also pointed out some limitations of the tool (some of them were indeed expected, because KTID current version is an initial version of the tool). In summary, considering the feedback provided by those two designers, the tool may be useful, and its use may be feasible, however it must be improved mainly in terms of its user interface and interaction aspects to be more user friendly. Moreover, it seems that the tool may be more useful for novice designers.

[1]https://coreui.io/vue-laravel/
[2]https://laravel.com/
[3]https://vuejs.org/

6 FINAL CONSIDERATIONS

In this paper, we propose to explore ontologies to support communication and knowledge sharing in HCI design. For that, we presented HCIDO, a reference ontology that provides a well-founded conceptualization of HCI design. HCIDO reuses concepts from HCI-ON [9] and SEON [24], connecting concepts from both areas and contributing to address the knowledge intersection between them. By providing a general conceptualization, HCIDO reduces semantic conflicts and helps communication and knowledge sharing.

We have used HCIDO in the development of KTID, a tool to support capture and sharing of knowledge embedded in design choices encoded in design specifications. The use of HCIDO facilitated KTID development by providing the domain conceptualization, which was used in the tool conception and to create its conceptual model. By doing so, we did not need to spend much time to understand the domain and create the tool conceptual model.

Although ontologies have been used in several domains, their use in HCI design needs to be further explored. This work gives a first step towards a set of envisioned ontology-based solutions to aid in HCI design. Currently, HCIDO does not have an operational version, i.e., we did not implement HCIDO in a language to be understood by computers (e.g., OWL). Therefore, HCIDO has been used at conceptual level and design time. Once we have an operational version of HCIDO, it can be used at run time and other solutions using ontology-related technologies (e.g., triple store and SPARQL queries) can be developed.

It is worthy clarifying that the KTID version referred in this paper is not a complete KM solution (e.g., it does not provide a robust curation to assess knowledge before make it available). We decided to provide just a simple evaluation in a five-star scale and focus on capturing and reusing knowledge to make KTID a simple solution. Moreover, HCI design involves human aspects, thus the problem addressed in this work may also be influenced by social, cultural, psychological and other factors. Therefore, the combination of ontologies and KM principles can contribute to solve communication and knowledge sharing issues in HCI design, but it should not be used as the only approach to handle that problem.

As future works, concerning KTID, we intend to improve it by considering the feedback provided by the two designers who used the tool. Then, we plan to evaluate the tool in practical settings, with a larger and heterogeneous population, in order to assess its capacity of supporting knowledge sharing and communication among

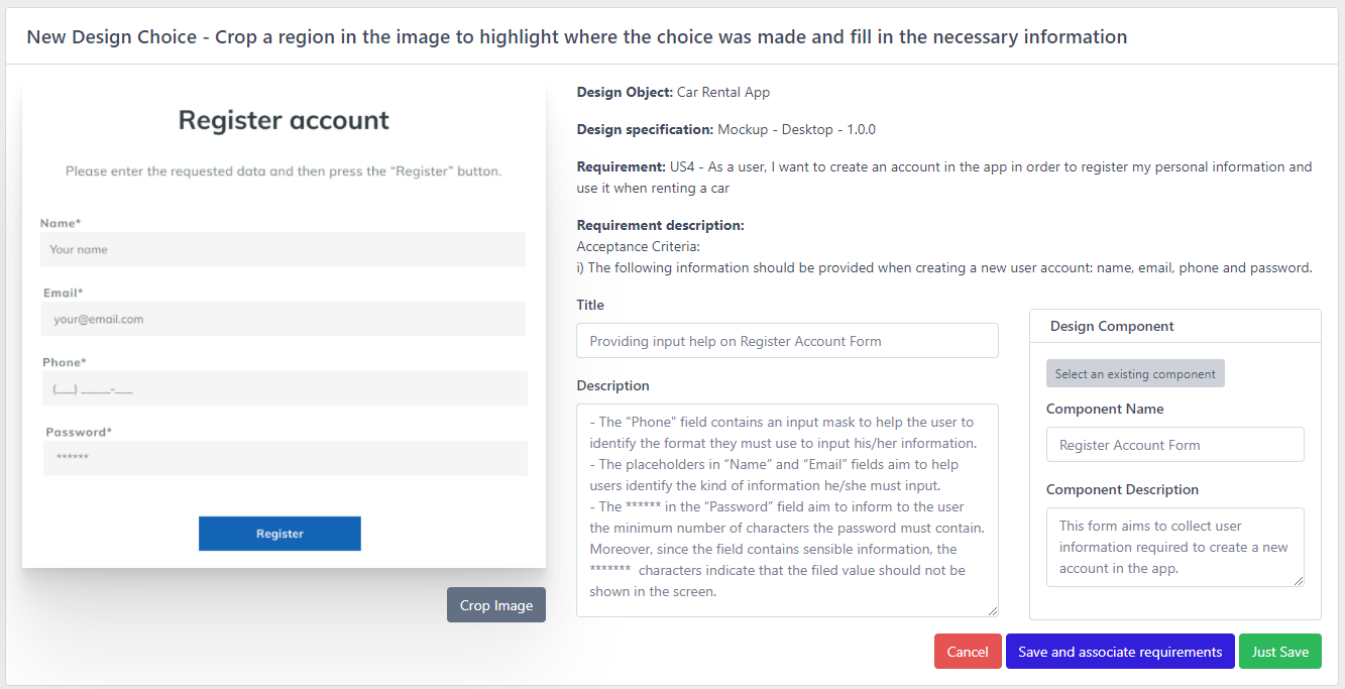

Figure 2: Describing a design choice in KTID.

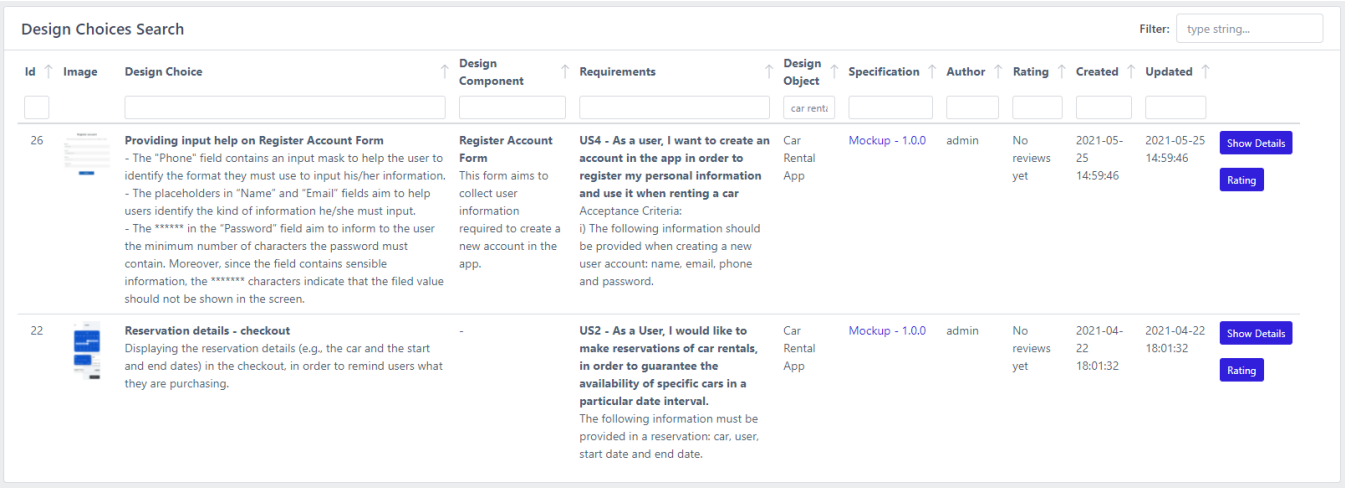

Figure 3: Searching for design choices recorded in KTID.

stakeholders with different backgrounds and to provide more general and conclusive results. In the future studies, we also intend to investigate the influence of using KTID on the creativity involved in the design process. Furthermore, we intend to implement features to assign requirements or design choices to developers and integrate KTID with software development management tools to provide integrated support to HCI design and software development activities. By doing this, we will explore deeper the connection between HCIDO and SEON, benefiting from the use of ontology networks to cover activities from the HCI design to the delivery of the interactive system to the client (e.g., involving implementation and test activities). As for HCIDO, we intend to explore its use in other applications to aid in HCI design, such as semantic documentation and semantic tools integration. We also plan to make the complete version of HCIDO available trough a website, so that other people can use it to produce their own ontology-based solutions to aid in HCI design.

REFERENCES

[1] Maxim Bakaev and Tatiana Avdeenko. 2010. Ontology to Support Web Design Activities in E-Commerce Software Development Process. In *ACIT - Information and Communication Technology*, Oleg I. Potaturkin and Yurii I. Shokin (Eds.). ACTAPRESS, Calgary,AB,Canada, 241–248. https://doi.org/10.2316/P.2010.691-075

[2] Maxim Bakaev and Martin Gaedke. 2016. Application of evolutionary algorithms in interaction design: From requirements and ontology to optimized web interface. In *2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*. IEEE, St. Petersburg, Russia, 129–134. https://doi.org/10.1109/EIConRusNW.2016.7448138

[3] David Benyon. 2014. *Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design* (3rd ed.). Pearson. 640 pages.

[4] Janez Brank, Marko Grobelnik, and Dunja Mladenić. 2005. A survey of ontology evaluation techniques. In *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*. Ljubljana, Slovenia, 166–170. https://doi.org/10.1.1.101.4788

[5] John Millar Carroll. 2014. Human Computer Interaction (HCI). In *The Encyclopedia of Human-Computer Interaction* (2nd ed.), Mads Soegaard and Rikke Friis Dam (Eds.). The Interaction Design Foundation, Aarhus, Denmark, Chapter 2, 21–61.

[6] Murillo Vasconcelos H. B. Castro, Simone Dornelas Costa, Monalessa P. Barcellos, and Ricardo de A. Falbo. 2020. Knowledge management in human-computer interaction design: A mapping study. In *23rd Iberoamerican Conference on Software Engineering, CIbSE 2020*.

[7] Murillo Vasconcelos H. B. Castro Castro. 2021. *Knowledge Management Solutions for Human Computer Interaction Design*. Master's thesis. Computer Science Department, Federal University of Espírito Santo (UFES).

[8] Catalin-Marian Chera, Wei-Tek Tsai, and Radu-Daniel Vatavu. 2012. Gesture ontology for informing Service-oriented Architecture. In *2012 IEEE International Symposium on Intelligent Control*. IEEE, Dubrovnik, Croatia, 1184–1189. https://doi.org/10.1109/ISIC.2012.6398257

[9] Simone Dornelas Costa, Monalessa Perini Barcellos, Ricardo de Almeida Falbo, and Murillo Vasconcelos Henriques Bittencourt Castro. 2020. Towards an Ontology Network on Human-Computer Interaction. In *Conceptual Modeling*, Gillian Dobbie, Ulrich Frank, Gerti Kappel, Stephen W Liddle, and Heinrich C Mayr (Eds.). Springer International Publishing, Cham, 331–341.

[10] Alan Dix, Janet E Finlay, Gregory D Abowd, and Russell Beale. 2004. *Human-Computer Interaction* (3rd ed.). Pearson Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 834 pages.

[11] Ricardo de Almeida Falbo. 2014. SABiO: Systematic Approach for Building Ontologies. In *CEUR Workshop Proceedings (CEUR Workshop Proceedings, Vol. 1301)*. CEUR-WS.org, Rio de Janeiro, Brazil.

[12] Nicola Guarino. 2014. Artefactual Systems, Missing Components and Replaceability. In *Artefact Kinds*. Springer International Publishing, Cham, 191–206. https://doi.org/10.1007/978-3-319-00801-1_11

[13] Giancarlo Guizzardi. 2005. *Ontological foundations for structural conceptual models*. Ph.D. Dissertation. University of Twente.

[14] Giancarlo Guizzardi. 2007. On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In *Proceedings of the Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV*, Olegas Vasilecas, Johan Edler, and Albertas Caplinskas (Eds.). IOS Press, Amsterdam, The Netherlands, The Netherlands, 18–39.

[15] ISO. 2019. *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. Standard ISO 9241-210:2019. International Organization for Standardization. 33 pages. https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-2:v1:en

[16] Eirini Myrgioti, Nick Bassiliades, and Amalia Miliou. 2013. Bridging the HASM: An OWL ontology for modeling the information pathways in haptic interfaces software. *Expert Systems with Applications* 40, 4 (mar 2013), 1358–1371. https://doi.org/10.1016/j.eswa.2012.08.053

[17] Abiodun Ogunyemi and David Lamas. 2014. Interplay between human-computer interaction and software engineering. In *Proceedings of the 9th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 1–10. https://doi.org/10.1109/CISTI.2014.6877024

[18] D.E. O'Leary. 1998. Enterprise knowledge management. *Computer* 31, 3 (mar 1998), 54–61. https://doi.org/10.1109/2.660190

[19] Heiko Paulheim and Florian Probst. 2013. UI2Ont—A Formal Ontology on User Interfaces and Interactions. In *Semantic Models for Adaptive Interactive Systems*. Springer, 1–24. https://doi.org/10.1007/978-1-4471-5301-6_1

[20] Michael Polanyi. 1966. *The Tacit Dimension*. Doubleday, Garden City, NY.

[21] Paul Ralph and Yair Wand. 2009. A Proposal for a Formal Definition of the Design Concept. In *Proceedings of the Design Requirements Engineering: A Ten-Year Perspective. Lecture Notes in Business Information Processing, vol 14.*, Kalle Lyytinen, Pericles Loucopoulos, John Mylopoulos, and Bill Robinson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 103–136. https://doi.org/10.1007/978-3-540-92966-6_6

[22] Yvonne Rogers, Helen Sharp, and Jenny Preece. 2011. *Interaction Design: Beyond Human-Computer Interaction* (3rd ed.). John Wiley & Sons, Chichester, United Kingdom. 585 pages.

[23] Ioana Rus and Mikael Lindvall. 2002. Knowledge management in software engineering. *IEEE Software* 19, 3 (may 2002), 26–38. https://doi.org/10.1109/MS.2002.1003450

[24] Fabiano Borges Ruy, Ricardo de Almeida Falbo, Monalessa Perini Barcellos, Simone Dornelas Costa, and Giancarlo Guizzardi. 2016. SEON: A Software Engineering Ontology Network. In *Knowledge Eng. and Knowledge Management*. Springer, 527–542.

[25] Kurt Schneider. 2009. *Experience and Knowledge Management in Software Engineering* (1st ed.). Springer Publishing Company, Incorporated, Heidelberg. 235 pages.

[26] Thiago Rocha Silva, Jean-Luc Hak, and Marco Winckler. 2017. A Formal Ontology for Describing Interactive Behaviors and Supporting Automated Testing on User Interfaces. *International Journal of Semantic Computing* 11, 04 (dec 2017), 513–539. https://doi.org/10.1142/S1793351X17400219

[27] Steffen Staab and Rudi Studer. 2004. *Handbook on Ontologies*. Springer Berlin Heidelberg, Berlin, Heidelberg. 660 pages. https://doi.org/10.1007/978-3-540-24750-0

[28] Rudi Studer, V.Richard Benjamins, and Dieter Fensel. 1998. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering* 25, 1-2 (mar 1998), 161–197. https://doi.org/10.1016/S0169-023X(97)00056-6

[29] Pablo Ribeiro Suàrez, Bernardo Lula Jùnior, and Marcelo Alves de Barros. 2004. Applying knowledge management in UI design process. In *Proceedings of the 3rd annual conference on Task models and diagrams - TAMODIA '04*, Pavel Slavik and Philippe Palanque (Eds.). ACM Press, New York, New York, USA, 113. https://doi.org/10.1145/1045446.1045468

[30] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi. 2012. *Ontology Engineering in a Networked World*. Springer Berlin Heidelberg, Berlin, Heidelberg. XII, 444 pages. https://doi.org/10.1007/978-3-642-24794-1

[31] Alistair G Sutcliffe. 2014. Requirements Engineering from an HCI Perspective. In *The Encyclopedia of Human-Computer Interaction* (2 ed.), Mads Soegaard and Rikke Friis Dam (Eds.). The Interaction Design Foundation, Aarhus, Denmark, Chapter 13, 707–760.