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

Murillo Vasconcelos Henriques Bittencourt Castro
murillo.castro@aluno.ufes.br
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department, UFES
Vitória, ES, Brazil

Monalessa Perini Barcellos
monalessa@inf.ufes.br
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department, UFES
Vitória, ES, Brazil

Ricardo de Almeida Falbo
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department, UFES
Vitória, ES, Brazil

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 to the quality of such systems. HCI design often involves people with different backgrounds (e.g., Arts, Software Engineering, Design). This makes knowledge transfer a challenging issue due to the lack of a common conceptualization about HCI design, leading to semantic interoperability problems, such as ambiguity and imprecision when interpreting shared information. 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 the use of ontologies to represent structured knowledge of HCI design and improve knowledge sharing in this context. We developed the Human-Computer Interaction Design Ontology (HCIDO), which is part of the Human-Computer Interaction Ontology Network (HCI-ON) and is connected to the Software Engineering Ontology Network (SEON). By making knowledge related to the HCI design domain explicit and structured, HCIDO helped us to develop KTID, a tool that aims to support capturing and sharing knowledge to aid in HCI design by allowing HCI designers to annotate information about design choices in design artifacts shared with HCI design stakeholders. Preliminary results indicate that the tool can be particularly useful for novice HCI designers.

CCS CONCEPTS

• **Software and its engineering → Designing software**.

KEYWORDS

HCI Design, User Interface, Knowledge Management, Ontology

ACM Reference Format:
Murillo Vasconcelos Henriques Bittencourt Castro, Monalessa Perini Barcellos, and Ricardo de Almeida Falbo. 2022. An Ontology to support Knowledge Management Solutions for Human-Computer Interaction Design. In *Proceedings of The 21st Brazilian Symposium on Software Quality (SBQS 2022).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

SBQS 2022, November 07–10, 2022, Curitiba, PR, Brazil

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

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 [8]. HCI design comprises practices, principles and knowledge from multiple fields, such as Ergonomics, Cognitive Science, Sociology, Human Factors and Computer Science [45]. 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 software product may be conflicting among different stakeholders, which hampers communication and knowledge transfer [8] [37]. Therefore, it is important to reach a consensual understanding on the meaning of terms related to HCI design 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 [12].

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) and support interface adaptation, among others [17]. However, there is still a need of properly understanding HCI design and its relation with other aspects of the software engineering, 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 developed the *Human-Computer Interaction Design Ontology* (HCIDO), a reference ontology that addresses 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 of HCI design and connects them to other aspects of the software engineering, providing an integrated view of software development. Moreover, considering KM principles that seek to transform individual and implicit knowledge into organizational and explicit knowledge, we 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. KTID was evaluated by two designers and the preliminary results indicate that the tool can be particularly useful for novice designers, who need to learn from previous experiences to create new designs. Experienced designers, in turn, may prefer a more creative and individual process, being more willing to record and share knowledge than to reuse knowledge recorded in the tool.

The remainder of this paper is organized as follows: Section 2 regards the research method adopted in this work and highlights the work contributions; Section 3 briefly presents the background for the paper and discusses some related work; Section 4 introduces HCIDO; Section 5 concerns KTID; and Section 6 concludes the paper.

2 RESEARCH METHOD

The research method adopted in this work followed the Design Science Research (DSR) paradigm [27]. It comprises the following steps [35]: (i) *Problem identification and motivation; (ii) Definition of 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 [27]: *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 first author when working on the development of interactive systems as a software engineer together with HCI designers. The 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 to manage knowledge in the HCI design of interactive systems.

Aiming to understand the subject in depth, we investigated the state of the art about knowledge management in HCI design through a systematic mapping [12]. 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 to 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 [13]. The survey results reinforced the lack of a consensual conceptualization about HCI design as a challenge and indicated that managing HCI design knowledge has been a concern that professionals have tried to deal with by using informal ways or simple tools.

Considering the identified problem, the gaps pointed by the systematic mapping, the survey results, and the capacity of ontologies to contribute to semantic interoperability and 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 and use it to develop a KM solution to aid HCI design. We advocate that the use of ontologies to establish a common conceptualization could help deal with communication and knowledge sharing problems in the HCI design context. However, the survey results showed that practitioners are not very familiar with ontologies. Therefore, we argue that the ontology-based KM solution should abstract the ontology to final users.

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) to provide an integrated view of HCI design into the software engineering as a whole, the ontology must consider aspects related from both HCI and Software Engineering (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 able to 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 and effective solution for modeling a problem and it is at the same time good (in terms of formal quality), usable and practicable [20]. In addition to the requirements to be met by the ontology, based on [21], 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 [17] and reuses concepts from SEON ontologies [39], particularly from the Software Design Reference Ontology (SDRO) [10], which was also developed in this work and reuses concepts from other SEON ontologies. To satisfy R3, HCIDO is organized into two sub-ontologies. To meet R4, we defined HCIDO by means of conceptual models (represented in the Unified Modeling Language) and textual descriptions. Concerning R5, we grounded HCIO in the Unified Foundational Ontology (UFO) [23]. As for R6, we followed SABiO (Systematic Approach for Building Ontologies) [21]. Then, in the *"Demonstration"* step, we used HCIDO as a reference model in the development of KTID, a tool that supports KM aspects in HCI design. During the *Evaluation* step, to evaluate HCIDO considering C1 and C2, we performed verification and validation activities, as suggested in SABiO [21]. To evaluate HCIDO considering C3, we performed a study in which two designers used and evaluated KTID.

Finally, the *"Communication"* step involves presenting the research results to the Academic and Industry communities. The results produced in this work were published in [9–13, 17]. The main contribution of the work is HCIDO, a reference ontology providing a well-founded conceptualization about HCI design that can be used to support interoperability and knowledge-related solutions. There are also other relevant contributions: (*i*) the systematic mapping that provides a panorama of the state of the art of KM in HCI design [12]; (*ii*) the survey that consolidate evidence about KM in HCI design practice [13]; (*iii*) SDRO, a reference ontology about software design, which served as the basis to HCIDO and is a contribution by itself, structuring knowledge about software design which can be used in interoperability and knowledge-related solutions in this context [10]; and (*iv*) KTID, a computational tool based on HCIDO that supports KM in the HCI design of interactive systems [11].

In this paper, our focus is on providing an overview of the research and introducing HCIDO. Further information about the other contributions can be found in [10–13].

3 BACKGROUND

3.1 HCI Design and Knowledge Management

HCI design focuses on how to design a system to support users to achieve their goals through the interaction with that system [45]. It is concerned with usability, user experience and other important attributes such as, accessibility and communicability. HCI design is user-centered, hence it is said User-Centered Design (UCD) [14]. UCD focuses on the use and development of interactive systems, with an emphasis on making products usable and understandable. It puts human needs, capabilities and behavior first, then designs the system to accommodate them. 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 [28]. HCI design is as an intensive knowledge process, requiring effective mechanisms to collaboratively create and support a shared understanding of the users, the system, its purposes, context of use and the design necessary for the user to achieve her goals. Therefore, HCI design could take advantage of knowledge management solutions.

According to [41], knowledge is a human specialty stored in people's minds, acquired through experience and interaction with their environment. 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 [32, 38, 41]. Knowledge helps software organizations to react faster and better, supporting more accurate and precise responses, which contributes to increasing software quality and client satisfaction [41].

When an organization implements KM, its experiences and knowledge are recorded, evaluated, preserved, designed and systematically propagated to solve problems [41]. Thus, KM addresses knowledge in its evolution cycle, which consists of creating, capturing, transforming, accessing and applying knowledge [38, 41]. In the Software Engineering context, it has been common to focus on KM solutions that deal with explicit knowledge and there is a need to focus also on tacit knowledge [5]. Both tacit and explicit knowledge are considered important for the Software Engineering community. However, one of the main challenges is the conversion of tacit knowledge into explicit knowledge [49].

3.2 Ontologies

An ontology is a formal and explicit specification of a shared conceptualization [44]. 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 agent is committed, either explicitly or implicitly, with one conceptualization [43].

According to [40], ontologies can be organized in a three-layered architecture: (i) *foundational ontologies* model the very basic and general concepts and relations that make up the world (e.g., objects, events, agents); (ii) *core ontologies* refine (i) by adding detailed concepts and relations in a specific area (e.g., organization, software process); and, (iii) *domain ontologies* describe a particular domain in reality (e.g., software organization, software testing) by specializing concepts from (i) or (ii).

Another important distinction differentiates ontologies as conceptual models, called *reference ontologies*, from ontologies as computational artifacts, called *operational ontologies* [24]. 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 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 [47]. Thus, an ON contributes to representing and growing knowledge of a domain by connecting several subdomains inside the ON or different domains, in a network of ONs. The ontology introduced

in this paper (HCIDO) is connected to two ONs. It is part of the Human-Computer Interaction Ontology Network (HCI-ON) [17], which contains ontologies addressing HCI subdomains (e.g., HCI phenomenon, HCI Evaluation), and is connected to the Software Engineering Ontology Network (SEON) [39], which includes ontologies addressing SE subdomains (e.g., Software Process, Software System, Software Requirements, Software Testing).

In the literature, there are some works using ontologies to support aspects related to HCI design. According to Paulheim and Probst [33], the main purposes for using ontologies in this context have been improving: (i) user interface visualization capabilities, (ii) user interface interaction possibilities, and (iii) user interface development process. Proposals aiming at (i) and (ii), often employ operational ontologies at run-time, sometimes combined with other tools (e.g., reasoners), to change the user interface or the system interaction (e.g., [29, 30]). On the other hand, approaches that aim (iii) often use ontologies at design time and the end user does not see the ontologies, nor interact with them. Examples are the use of ontologies to support the creation of metamodels in model driven approaches (e.g., [46]), or the annotation and classification of user interface components in repositories (e.g., [25, 26]).

With regard to the scope addressed by ontologies used in the HCI design context, they have usually focused either on the interaction between user and system (e.g., [42]) or on the user interface and its components (e.g., [34]). Hence, they have not focused on describing the design of the human-computer interaction itself. The ontologies that address the HCI design itself, have focused on HCI design in specific contexts, such as web design [1, 2], design for haptic devices [31] and design for gesture based interactions [15]. Therefore, none of the ontologies we found by carrying out a systematic review of the literature [16] 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. Lastly, none of them connect HCI design to other aspects of the software engineering and, thus, do not provide an integrated view of HCI design and software development.

4 THE HUMAN-COMPUTER INTERACTION DESIGN ONTOLOGY (HCIDO)

The investigation of the state of the art [12] and the state of the practice [13] about KM in HCI design 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 [44]. Hence, this section introduces the Human-Computer Interaction Design Ontology (HCIDO), which aims at establishing a common conceptualization of HCI design of interactive systems, and that can be used for communication purposes as well as to solve interoperability and knowledge-related problems.

HCIDO is a networked domain ontology of HCI-ON [17] and reuses concepts related to HCI and SE by specializing other ontologies from HCI-ON and SEON [39], respectively. By doing that, HCIDO connects HCI design to other aspects of the software engineering (e.g., choices made in the HCI design are connected to the software requirements that motivated them). Figure 1 presents an overview of HCI-ON and SEON showing some of their ontologies and positioning HCIDO and SRDO, the ontologies developed in this work (they are highlighted by thicker black lines in the figure). The ONs architecture follows the three-layers defined by Scherp *et al.* [40] (see Section 3). As all the ontologies from SEON and HCI-ON, HCIDO is grounded in UFO [23]. In the figure, each circle (network's node) represents an ontology. Red arrows (directed arcs) represent relationships between HCI-ON and SEON ontologies. Dependencies between HCI-ON and SEON core ontologies are denoted by red solid arrows, while between domain ontologies are denoted by red dotted arrows. A dependency relation between two ontologies means that the source ontology reuses concepts from the target ontology.

Figure 2 presents HCIDO architecture (as a UML package diagram) by extracting from Figure 1 the ontologies directly reused by HCIDO and detailing dependencies and modules. In the figure, red double-dashed lines separate core and domain layers. For simplification reasons, the grounding of HCIDO in UFO is not discussed in this paper.

From SEON, we highlight the *Software Design Reference Ontology (SDRO)* [10], which was also developed in this work and addresses aspects of software design, connecting them to other SE aspects, such as requirements, code and testing [7, 19]. SDRO describes the mental and physical elements involved in the design of software systems and the relations between them [10]. Here, the term "physical" is borrowed from other works [3, 22, 36] referring to the perception of something through the senses. However, considering that software is abstract, there are differences in the way it is perceived when compared to physical objects like a chair or a car.

Hence, SDRO conceptualization helps highlight what makes software and software design unique when compared to other fields involving design: there is a large gap between what is produced as the result of the design effort and what is perceived by the user in his/her interaction experience [10]. The result of the design effort, a software system constituted by code [19], does not interact directly with the user as does a car or a house, for example. It must be loaded in a computer system and then executed, so the user can interact with the result of this execution. 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 [48]. To reduce the gap between software design and user experience, it is essential to have a holistic view of design and look at software and its construction as a whole. HCIDO aims to help reduce this gap by merging design aspects from Software Engineering and HCI domains.

Considering that HCIDO addresses a particular type of software design, it reuses the distinction between mental and physical aspects from SDRO. For that, it reuses concepts from SDRO such as Software Designer, Software Design Choice, Software Design Specification, Software Designer Mental Moment, among others. Moreover, HCIDO is focused on characterizing specific aspects of

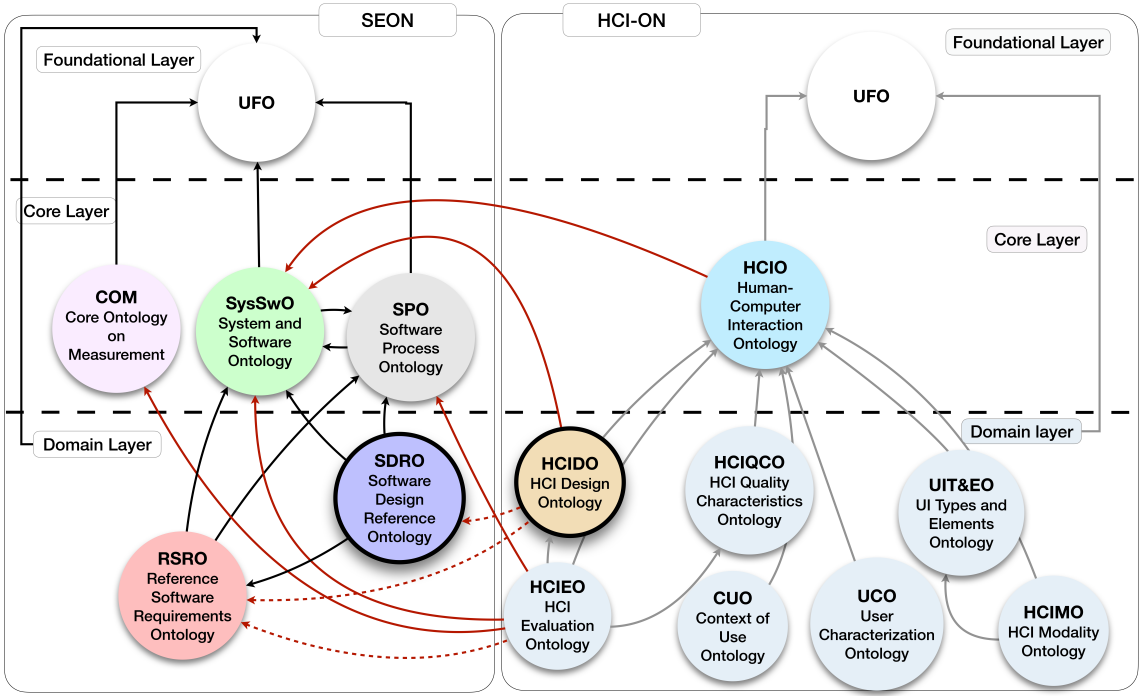

Figure 1: Placement of HCIDO and SDRO in HCI-ON and SEON architectures.

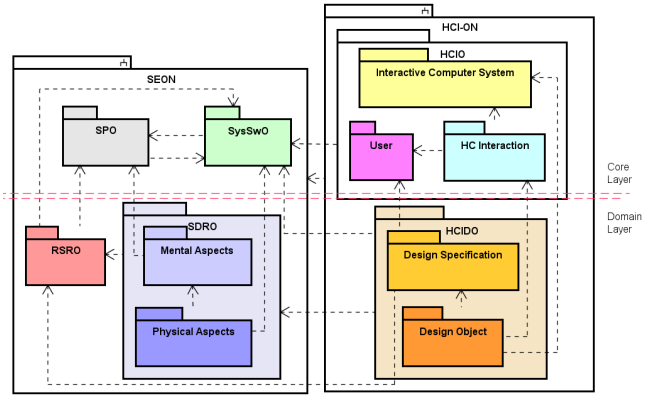

Figure 2: Figure 2: HCIDO architecture.

design when designing HCI of interactive systems. Thus, it specializes SDRO to the HCI domain by reusing HCI concepts from HCIO [17, 18], the Human-Computer Interaction Ontology, which addresses this phenomenon considering the participation of the user and the system. HCIO includes concepts such as User, Interactive Software System, User Interface, among others. By specializing SDRO and connecting the specialized concepts to HCIO concepts, HCIDO connects the SE and HCI perspectives of HCI design.

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 the HCIDO conceptual model is presented in Figure 3. As shown in Figure 2, HCIDO is modularized into two sub-ontologies: the *Design Specification* sub-ontology and the *Design Object* sub-ontology. The former focuses on the artifacts created to specify the design object, while the latter is devoted to the design object itself (i.e., the interactive system) and its components. Both reuse mental and physical-related concepts from SDRO, in the sense that both mental and physical aspects are involved in the creation of the design specifications or the design object itself. Due to space limitation, the figure does not include concepts related to mental aspects. In the figure, the concepts are presented in the same color used in Figure 2, indicating the ontologies to which they belong. Blue lines indicate relationships between HCIDO and other ontologies from the networks. After the figure, we present a brief description of HCIDO concepts. 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*. The complete and detailed models and descriptions of HCIDO, including mental aspects, a discussion about the grounding in UFO and the specialization from other ontologies concepts, plus examples of the concepts (i.e., instances) can be found in [9].

To better understand HCIDO concepts, it is necessary, first, to understand some concepts from SEON [39]. A *Person Stakeholder* is a person interested or affected by software development activities or their results. *Software Artifacts* are objects intentionally

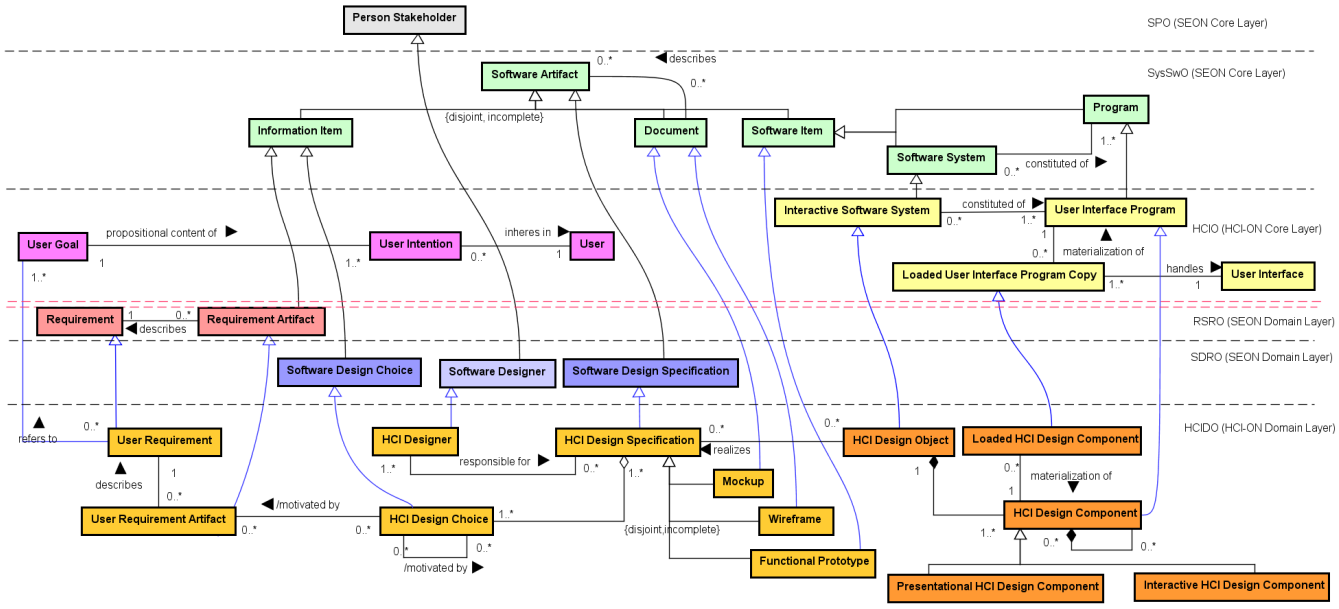

Figure 3: Fragment of HCIDO conceptual model.

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 fights 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 from SEON (i.e., related to SE), it is also necessary to understand some HCI core concepts from HCI-ON [17]. An *Interactive Software System* is a *Software System* constituted (among others) of *User Interface Programs*, which are *Programs* that handle the *User Interface* through their materialization as *Loaded User Interface Program Copies* (i.e., copies of the program loaded in the computer system memory). The *User Interface* comprises all parts of the computer system that users have contact with, physically, perceptually or conceptually [4]. *Users* interact with the system to achieve *User Goals*, which represent needs intended to be satisfied by the system and are the propositional content of *User Intentions* that inhere in a *User*.

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 3, 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

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. **HCI Design Components** are materialized as **Loaded HCI Design Components**, which are *Loaded User Interface Program Copies* (i.e., copies of programs that deal with user interface aspects loaded in the memory of an interactive computer system).

As we explained before, although not shown in Figure 3, 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 3. 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 [21], 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 [9], as suggested in [6]. After that, we used HCIDO as a basis to develop a tool to help knowledge capture and sharing in HCI design.

5 KTID: A COMPUTATIONAL TOOL TO SUPPORT KM CAPTURE AND SHARING IN HCI DESIGN

As previously discussed, 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 concepts and relations from HCIDO conceptual model to create KTID structural 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 structural 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 4 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 pieces of information have different semantics, they are syntactically represented in the same way in the mockup, relying on a

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

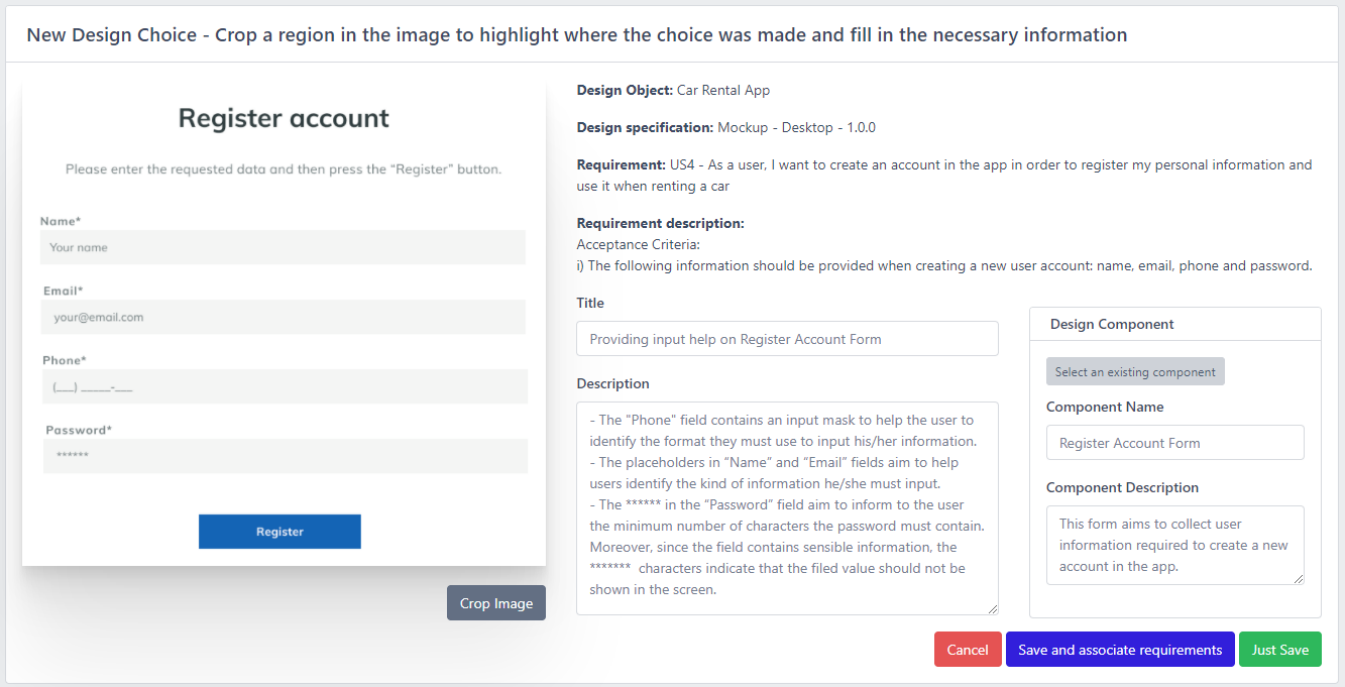

Figure 4: Recording a design choice in KTID.

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 5 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.

As a preliminary evaluation, we carried out a study in which 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 when he created the 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 aimed wireframe. He said that he could record his knowledge to future use, but he pointed out that this may promote some kind of standardization and prevent other designers from being more creative. In summary, the feedback provided by the two designers provides a preliminary indication that the tool is useful and its use is feasible. However, it should be improved to be more user friendly. Moreover, it seems that the tool may be more useful for novice designers.

6 FINAL CONSIDERATIONS

In the work described in this paper, we explored the combination of ontologies and ontology networks with KM to potentialize knowledge capture and reuse in the context of the HCI design of interactive systems. The main objective of this work was to propose a well-founded conceptualization of HCI design to support KM solutions to aid in HCI design of interactive systems. Thus, we developed SRDO, which addresses software design by considering both mental and physical aspects, and used it to develop HCIDO, a reference ontology that provides a well-founded conceptualization of HCI design. HCIDO reuses concepts from HCI-ON [17] and SEON [39], 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. Moreover, it can be used to build semantic interoperability and knowledge-based solutions.

This work provides contributions to both the state of the art and the state of the practice. SDRO and HCIDO are contributions to the state of the art since they structure knowledge of the software design and HCI design domains, respectively. Besides physical

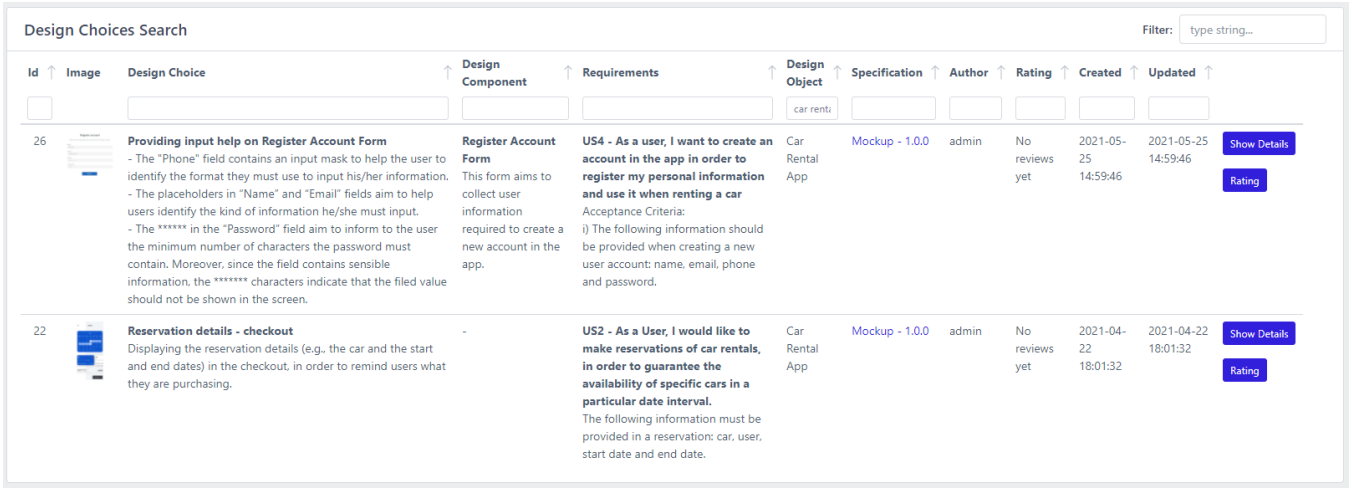

Figure 5: Searching for design choices recorded in KTID.

aspects, they also address mental design aspects, which are relevant to a better understanding of how ideas that come up in the designers' mind are materialized into artifacts produced during the software development process (e.g., mockups, wireframes). The systematic mapping about KM in HCI design is also a contribution to the state of the art, providing a panorama about the research topic. Regarding the state of the practice, the survey carried out with practitioners provides knowledge about the usage of KM in HCI design practice. Furthermore, KTID also contributes as a practical example of using HCIDO in the development of KM solutions to support HCI design and can be evolved to support the creation of a repository of HCI design decisions and components that can be reused in future projects.

We 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 structural model.

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 making it available). In this work, 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. KTID can be evolved to be a more robust solution and help create a repository of HCI design components (associated to their rationale) that can be reused in future design solutions. 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. However, we also need to point out that 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 knowledge sharing issues in HCI design, but it should not be used as the only approach to handle that problem.

As future work, 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 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.

REFERENCES

[1] Maxim Bakaev and Tatiana Avdeenko. 2010. Ontology to Support Web Design Activities in E-Commerce Software Development Process. August 2015 (2010). https://doi.org/10.2316/P.2010.691-075
[2] M. Bakaev and M. Gaedke. 2016. 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.* 129–134.
[3] Alex Baker and André van der Hoek. 2006. *Examining Software Design from a General Design Perspective.*
[4] D Benyon. 2013. Designing interactive systems: a comprehensive guide to HCI, UX and interaction design.
[5] Finn Olav Bjørnson and Torgeir Dingsøyr. 2008. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology* 50, 11 (2008), 1055–1068.
[6] Janez Brank, Marko Grobelnik, and Dunja Mladenić. 2005. A survey of ontology evaluation techniques. In *In In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005).*
[7] Ana Christina de Oliveira Bringuente, Ricardo de Almeida Falbo, and Giancarlo Guizzardi. 2011. Using a Foundational Ontology for Reengineering a Software Process Ontology. *Journal of Information and Data Management* 2, 3 (2011), 511–526. https://doi.org/10.5753/jidm.2011.1424

[8] John Millar Carroll. 2014. *Human Computer Interaction (HCI)* (2nd ed.). The Interaction Design Foundation, Chapter 2, 21–61.

[9] Murillo Vasconcelos Henriques Bittencourt Castro. 2021. *Knowledge Management Solutions for Human Computer Interaction Design.* Ph. D. Dissertation. Master Thesis, Graduate Program in Informatics, Federal University of Espírito Santo (UFES), Vitória - ES, Brazil.

[10] Murillo Vasconcelos Henriques Bittencourt Castro, Monalessa Perini Barcellos, and Ricardo de Almeida Falbo. 2021. An Ontological View of Design in the Software Context. In *14th Seminar on Ontology Research in Brazil (ONTOBRAS).* CEUR Workshop Proceedings.

[11] Murillo Vasconcelos Henriques Bittencourt Castro, Monalessa Perini Barcellos, Ricardo de Almeida Falbo, and Simone Dornelas Costa. 2021. Using Ontologies to aid Knowledge Sharing in HCI Design. In *XX Brazilian Symposium on Human Factors in Computing Systems (IHC'21).* ACM, New York, NY, USA. https://doi.org/10.1145/3472301.3484327

[12] Murillo Vasconcelos Henriques Bittencourt Castro, Simone Dornelas Costa, Monalessa Perini Barcellos, and Ricardo de Almeida Falbo. 2020. Knowledge management in human-computer interaction design: A mapping study. In *23rd Iberoamerican Conference on Software Engineering, CIbSE 2020.* in press.

[13] Murillo Vasconcelos Henriques Bittencourt Castro, Simone Dornelas Costa, Monalessa Perini Barcellos, and Ricardo de Almeida Falbo. 2022. Investigating Knowledge Management in Human-Computer Interaction Design. *Journal of Software Engineering Research and Development* 10, SE-Review (Mar 2022), 4:1–4:20. https://doi.org/10.5753/jserd.2021.1878

[14] Adriana Chammas, Manuela Quaresma, and Cláudia Mont'Alvão. 2015. A Closer Look on the User Centred Design. *Procedia Manufacturing* 3 (2015), 5397–5404. https://doi.org/10.1016/j.promfg.2015.07.656

[15] C Chera, W Tsai, and R Vatavu. 2012. Gesture ontology for informing Service-oriented Architecture. In *Int. Symposium on Intelligent Control.* IEEE, 1184–1189.

[16] Simone Dornelas Costa, Monalessa Perini Barcellos, and Ricardo de Almeida Falbo. 2021. Ontologies in Human-Computer Interaction: A Systematic Literature Review. *Applied Ontology* 16, 4 (2021), 421–452. https://doi.org/10.3233/AO-210255

[17] 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, 331–341.

[18] Simone Dornelas Costa, Monalessa Perini Barcellos, Ricardo de Almeida Falbo, Tayana Conte, and Káthia Marçal Oliveira. 2022. A core ontology on the Human–Computer Interaction phenomenon. *Data & Knowledge Engineering* 138 (2022). https://doi.org/10.1016/j.datak.2021.101977

[19] Bruno Borlini Duarte, Andre Luiz De Castro Leal, Ricardo De Almeida Falbo, Giancarlo Guizzardi, Renata S.S. Guizzardi, and Vítor E. Silva Souza. 2018. Ontological foundations for software requirements with a focus on requirements at runtime. *Applied Ontology* 13, 2 (2018), 73–105. https://doi.org/10.3233/AO-180197

[20] Mathieu d'Aquin and Aldo Gangemi. 2011. Is there beauty in ontologies? *Applied Ontology* 6 (2011), 165–175. https://doi.org/10.3233/AO-2011-0093

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

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

[23] Giancarlo Guizzardi. 2005. *Ontological foundations for structural conceptual models.* Ph. D. Dissertation. Telematica Instituut / CTIT.

[24] Giancarlo Guizzardi. 2007. 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.* IOS Press, 18–39.

[25] Hans-Jörg Happel, Axel Korthaus, Stefan Seedorf, and Peter Tomczyk. 2006. KOntoR: An Ontology-enabled Approach to Software Reuse. 349–354.

[26] Scott Henninger, Mohamed Keshk, and Ryan Kinworthy. 2004. Capturing and Disseminating Usability Patterns with Semantic Web Technology. (2004).

[27] Alan R Hevner. 2007. A three cycle view of design science research. *Scandinavian journal of information systems* 19, 2 (2007), 4.

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

[29] Andrea E Kohlhase and Michael Kohlhase. 2009. Semantic Transparency in User Assistance Systems. In *Proceedings of the 27th ACM International Conference on Design of Communication (SIGDOC '09).* Association for Computing Machinery, 89–96. https://doi.org/10.1145/1621995.1622013

[30] Marina Kultsova, Anastasiya Potseluico, Irina Zhukova, Alexander Skorikov, and Roman Romanenko. 2017. A Two-Phase Method of User Interface Adaptation for People with Special Needs. In *Creativity in Intelligent Tech and Data Science.* Springer, 805–821.

[31] 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 (2013), 1358–1371.

[32] Daniel E O'Leary. 1998. Enterprise Knowledge Management. *Computer* 31, 3 (Mar 1998), 54–61. https://doi.org/10.1109/2.660190

[33] Heiko Paulheim and Florian Probst. 2010. Ontology-Enhanced User Interfaces: A Survey. *Int. J. Semantic Web Inf. Syst.* 6 (2010), 36–59. https://doi.org/10.4018/jswis.2010040103

[34] Heiko Paulheim and Florian Probst. 2013. *UI2Ont—A Formal Ontology on User Interfaces and Interactions.* 1–24. https://doi.org/10.1007/978-1-4471-5301-6_1

[35] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. 2007. A design science research methodology for information systems research. *Journal of management information systems* 24, 3 (2007), 45–77.

[36] Paul Ralph and Yair Wand. 2009. A Proposal for a Formal Definition of the Design Concept. In *Design Requirements Eng.: A Ten-Year Perspective.* Springer, 103–136.

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

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

[39] 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.

[40] Ansgar Scherp, Carsten Saathoff, Thomas Franz, and Steffen Staab. 2011. Designing core ontologies. *Applied Ontology* 6, 3 (2011), 177–221.

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

[42] 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, 4 (Dec 2017), 513–539. https://doi.org/10.1142/S1793351X17400219

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

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

[45] Alistair G Sutcliffe. 2014. *Requirements Engineering from an HCI Perspective* (2 ed.). The Interaction Design Foundation, Chapter 13, 707–760.

[46] 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, 113–120. https://doi.org/10.1145/1045446.1045468

[47] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi. 2012. *Ontology Engineering in a Networked World.* Springer.

[48] Richard N Taylor and Andre van der Hoek. 2007. Software Design and Architecture: The once and future focus of software engineering. In *Future of Software Engineering (FOSE '07).* IEEE, 226–243. https://doi.org/10.1109/FOSE.2007.21

[49] Shanmuganathan Vasanthapriyan, Jing Tian, and Jianwen Xiang. 2015. A Survey on Knowledge Management in Software Engineering. In *2015 IEEE International Conference on Software Quality, Reliability and Security - Companion.* 237–244. https://doi.org/10.1109/QRS-C.2015.48