Design Pattern as a Bridge Between Problem-Space and Solution-Space

Jan Marco Leimeister, Ernestine Dickhaut, and Andreas Janson

Abstract Designing novel technologies provide challenges to developers. To support developers in designing these technologies, design knowledge must be codified and made applicable for the future. In systems development, design patterns provide proven solutions to solving recurring problems. They contain templates for describing design information, often in tabular form, and are established tools for making complex knowledge accessible and applicable. Design patterns play a critical role in both practice and research in finding potential solutions. For researchers, patterns can provide a method for codifying design knowledge for future research. For practitioners, design patterns provide established solutions to recurring problems. By applying them in a particular context, the pattern represents elements of both the problem-space and the solution-space, providing an opportunity to bridge the gap between the two spaces. Due to the abstraction of design patterns, they can be used for different application scenarios. The preparation of the design knowledge in the design pattern is a critical step to support the user in the best possible way, that determines the usefulness of the pattern.

Keywords Design pattern · Design knowledge · Design science research

J. M. Leimeister (✉)
Institute of Information Management, University of St. Gallen, St. Gallen, Switzerland

Information Systems, Interdisciplinary Research Center for IS Design (ITeG), University of Kassel, Kassel, Germany
e-mail: janmarco.leimeister@unisg.ch

E. Dickhaut
Information Systems, Interdisciplinary Research Center for IS Design (ITeG), University of Kassel, Kassel, Germany
e-mail: ernestine.dickhaut@uni-kassel.de

A. Janson
Institute of Information Management, University of St. Gallen, St. Gallen, Switzerland
e-mail: andreas.janson@unisg.ch

S. Aier et al. (eds.), *Engineering the Transformation of the Enterprise*,
https://doi.org/10.1007/978-3-030-84655-8_9

1 Introduction

In information systems (IS) research, the accumulation of design knowledge is becoming increasingly important for research and practice [1]. In particular, the Design Science Research (DSR) paradigm, which is widely used to design and develop technologies [2], focuses on the development and evaluation of new technologies, applying rules and concepts such as design theories and principles that can be used to map and support design processes [3]. However, at first, it is necessary to take a closer look at what design knowledge is and how it is generated in DSR projects. The ability to generate and use knowledge has become an important quality characteristic for design-oriented research [4]. When we study the understanding and use of design knowledge in IS research, the focus is on harnessing design knowledge for the future. Vom Brocke et al. [5, p.6] emphasize "the goal of DSR is to generate knowledge about how to effectively build innovative solutions to important problems". Good design should not be used only for a "single success story" [6]. Reusability and learning from design knowledge are critical to the success of DSR projects and beyond. This raises the problem of how acquired (design) knowledge should be codified so that others can use the knowledge to solve design problems in depth.

Contextual knowledge is often necessary to better classify the problem and thus identify and develop the right solution approach to solve a problem. To achieve this, the problem context must initially be understood, the problem identified and classified, and then the appropriate solution developed. To be able to solve problems in development, support is needed that codifies extensive design knowledge in a practical manner and helps the user to find a suitable solution for the problem at hand. For this purpose, design knowledge must be codified in a practical one.

This paper introduces the approach of design patterns for the codification of design knowledge. The peculiarity of design patterns is that they combine content that provides a solution direction with information from the problem context [6]. Thus, design patterns provide elements from both the problem-space and the solution-space, allowing the user to find creative solutions suitable for the problem at hand with the help of codified design knowledge. To this end, design patterns differ from other approaches to codify design knowledge, such as design principles, and the distinctive features of design patterns in unifying elements from problem-space and solution-space are elaborated.

2 The Concept of Knowledge in the Background of Design-Oriented Research

Before we can consider the specificity of codifying design knowledge in design patterns, we first have to take a closer look at what knowledge is generally and how design knowledge is distinguished from knowledge.

Knowledge comprises information that is gathered through the interpretation of experiences. Therefore, knowledge is built up through interaction with the world and is organized and stored in the mind of an individual. Two forms of knowledge can be distinguished:

1. *Tacit knowledge exists* unconsciously in a person's mind without the need to put it into words.
2. *Explicit knowledge* can be communicated to others and recorded in written documents and procedures.

While explicit knowledge can be easily transferred, other types of knowledge, such as tacit knowledge, are difficult to transfer [7]. Knowledge is created by individuals and becomes valuable by being passed on to other individuals [8]. However, to make the knowledge usable for the future and accessible to others, it must be captured and codified [9]. Challenges arise particularly from the application of theoretical knowledge in a practical context [10]. To ensure that this knowledge has added value in practice, it must be individually adapted to the respective context.

In addition to the distinction between tacit and explicit knowledge, knowledge can be divided into know-how, know-why, and know-what [11]. Know-how knowledge is acquired through "learning-by-doing", i.e. in the execution of the activity. Second, know-why is based on rules and techniques learned through training. Third, know-what is an important way in which knowledge is generated [11]. To acquire know-what knowledge, information about the problem context and the need to implement the solution must be provided.

Knowledge generation is a process in which individuals cross the boundary of the old into a new self by acquiring new knowledge [8]. In the process, new conceptual artifacts and structures for interactions emerge that offer possibility. Knowledge influences how reality is viewed. The same reality can be viewed differently depending on the knowledge one has. The context of the situation is important to generate knowledge because the context helps to relate and classify the information.

Knowledge generation starts with socialization, which is the process of transforming new tacit knowledge through shared experiences in daily social interaction. Tacit knowledge is difficult to formalize, thus, it is often time and space specific. The knowledge can only be acquired through shared experience [1].

Design knowledge is a special form of knowledge, namely knowledge about the design of a system including the associated methods and constructs [2]. A person acquires this knowledge, for example, through experience in designing systems or through education. Design knowledge is gained from creative insights as well as trial-and-error processes and therefore does not usually have a close deductive relationship to existing scientific knowledge [12]. It exists initially only as tacit knowledge in the person's mind. The experiential space, and thus the existing design knowledge of a person, varies depending on the person's level of experience. By actively applying the tacit knowledge, an individual solution can be found. Design knowledge should ideally be reusable for similar problems. However, challenges arise in an application when problems at hand are very general and the design knowledge must be abstract yet actionable enough [13].

Design knowledge is characterized by vagueness, hierarchy, and links to other knowledge-bits, which makes it difficult to share [14]. In system development, the sharing of design knowledge is an important step towards developing high-quality systems with high user acceptance. In large software projects, knowledge does not exist in the head of a single person [14] but is based on the shared knowledge of several people. Thus, implicit design knowledge should be transferred into explicit knowledge and made accessible to other people [15]. In practice, internal company wikis are often used to organize knowledge transfer and design patterns for this purpose. We will look at these challenges in the next chapter, where processes of codifying design knowledge will be addressed.

3 Sharing Design Knowledge

3.1 Externalization of Design Knowledge

The SECI model (socialization—externalization—combination—internalization) by Nonaka and Takeuchi is a concept that describes the emergence of organizational knowledge [2]. As a standard model for knowledge generation, transfer, and development of knowledge, the SECI model forms a basis for describing the explication of tacit knowledge into explicit knowledge (see Fig. 1).

The model classifies knowledge into explicit knowledge, which is objective, codified, transferable, and formal, and tacit experiential knowledge, which is difficult or impossible to convey and is only very inadequately codified. Tacit knowledge

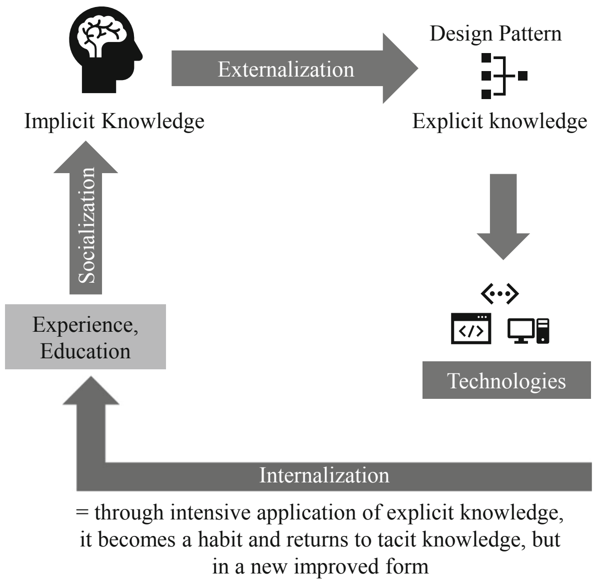

Design Pattern
Externalization
Implicit Knowledge
Explicit knowledge
Socialization
Experience, Education
Technologies
Internalization
= through intensive application of explicit knowledge, it becomes a habit and returns to tacit knowledge, but in a new improved form

Fig. 1 Externalization of design knowledge (based on the SECI model) [8]

is based on experiences, culture, emotions, and values and is manifested in methodological and social competencies more than in qualifications, i.e. also in action routines and procedures, but also convictions, beliefs, and culturally codified schemata. The explication of an implicit context is an essential prerequisite for the creation of new knowledge.

Socialization is the exchange of experience through both horizontal and vertical communication. In the exchange of experience, knowledge is developed through conversation, at a conference, or through imitation. According to the SECI model, the transformation from unconscious to conscious knowledge is called externalization. By including images, metaphors, or models, implicit knowledge is externalized in such a way that it can be understood by others. In combination (combination = explicit-to-explicit), existing explicit knowledge is combined with other knowledge content to form new, explicit knowledge. Different contents can be transformed by media, from analog to digital, and expanded by adding other contexts.

The reverse process from conscious to unconscious knowledge is called internalization. In this process, explicit knowledge is transformed into tacit knowledge, which means that experiences and knowledge gained through previous socialization, externalization, or combination are integrated into individual mental models.

The SECI model can be applied to the process of codifying design knowledge through design patterns. Design knowledge is initially tacit and difficult for other people to grasp in its form. By codifying design knowledge in design patterns, it is codified and recorded in a communicable form of representation. Now the knowledge can be picked up by other people and be used as a set of rules. Design knowledge can exist in many disciplines. For example, legal design knowledge differs from design knowledge in DSR. The legal expert brings knowledge about legal requirements to design novel technologies [16]. Nevertheless, legal design knowledge can also be codified into design patterns (externalization) [17]. The codified legal design knowledge can be used by others. However, there is special caution required in the way the design knowledge is represented if people outside the domain are to use the externalized design knowledge. A developer may have a large repertoire of solutions to design software, but usually little legal knowledge.

3.2 *Codification Approaches of Design Knowledge*

Since knowledge is often vaguely formulated, the codification of design knowledge requires special methods. There are already various approaches that deal with methods for codifying design knowledge to be able to capture it for the future and to share it with others (see Table 1).

Cheat sheets are a common method for capturing design knowledge, especially in system development. A large amount of knowledge is stored in a small space, usually on a "one-pager" [25]. This allows the user to obtain a large amount of information briefly and thus find a solution to any problem as quickly as possible

Table 1 Overview of various methods for codifying design knowledge

Categorization of knowledge (according to [3])	Method for codifying design knowledge	Source
Know-how	Knowledge maps	[18]
	Mind maps	[19]
	Conceptual maps	[18]
	Wikis	[20]
	Prototypes	[21]
	Design principles	[22]
	Design patterns	[23]
Know-why	Cheat sheets	[24]
	Prototypes	[21]
	Design patterns	[23]
Know-what	Wikis	[20]
	Design principles	[22]
	Design Patterns	[23]

[26]. In particular, the development of cheat sheets is a crucial way to retrieve knowledge and bring it to the point from a learning perspective [24].

Knowledge maps or similar approaches such as mind maps or conceptual maps are based on the idea of representing knowledge as it is stored in memory. In comparison to cheat sheets, they use a graphical representation and visualize relationships between individual elements using connections and arrows [18]. Knowledge maps have proven useful in identifying areas where are gaps in knowledge, resources, and knowledge flow. These approaches are especially used when knowledge is to be captured but not shared. Wikis are created for knowledge transfer, especially in teams and organizations. They focus on the transfer of individual knowledge [20].

Wikis have the advantage of an overview page that allows users to find exactly the information they need. In practice, wikis often replace exchanges between individuals within an organization. The way knowledge is codified within the wiki varies depending on the author of the contribution. Knowledge transfer between IT professionals often occurs through information systems. This often includes additional information found in system documentation and user training materials [27].

Prototypes are often used for communication between developers and other disciplines. They allow both sides to demonstrate requirements and possible functionalities in a practical way [21]. They provide a cost-effective demonstration of the system and offer the opportunity to contribute to expertise [28]. These approaches do not provide a solution to the challenge of codifying design knowledge in such a way that it can be understood by disciplines outside the domain. Facts such as lack of structure, use of technical terms, and incompleteness lead to low use of these tools. It must be ensured that design knowledge is formulated in a clear, unambiguous, and accessible language and is free of inconsistencies and contradictions [10].

In system development, design patterns are a solution to solve recurring problems and challenges [23]. Design patterns are established tools for making complex knowledge accessible to system developers. The advantage of patterns in comparison to design principles, for example, is that they cover all three forms of knowledge while providing an established approach to development practice. In the literature, patterns are often referred to as "templates" of established solutions for frequently recurring problems in system development.

4 Design Science Research and Design Patterns

Although the IS discipline has extensive experience in digitizing and designing sociotechnical artifacts, the underlying design knowledge is not systematically accumulated across different settings and projects and therefore cannot be transferred and reused in new contexts [29].

Design patterns originally come from the field of architecture and were established by the work around Alexander et al. [30], in which architects are supported in the design of houses and the planning of cities by design patterns. In design, architects regularly face recurring problems in which proven solutions can be applied. Proven solutions do not add value to others until they are shared on a sufficient scale. Thus, design patterns are used to capture proven solutions to recurring problems and make them usable for the future.

After design patterns have revealed themselves to be useful in architecture, many different disciplines adopt the idea of design patterns and transferred it to various disciplines. In system development, design patterns have become established primarily through the work of the so-called "Gang of Four (GoF)" around Gamma et al. [31].

Design patterns represent abstract and thus generally applicable and reusable solutions for recurring problems. "Best practices" are codified in design patterns and are made usable for the future. For this purpose, a design pattern offers a kind of "template", which is structured in the same way for all patterns. They do not present innovative solutions and do not "reinvent the wheel" but build on proven solutions. At the same time, a design pattern offers approaches to solutions for many problems. Thus, design patterns address one of the problems identified by Brocke et al. [5] in the reuse of design knowledge by codifying design knowledge in design patterns in an abstract way.

Petter et al. [32] see four crucial phases in the life cycle of design patterns (see Fig. 2). In the first phase, the development of the pattern, both domain knowledge from the corresponding domain and theories from the literature together with existing "best practice" solutions flow into the development. Thereupon, the pattern is used in the second phase and comes to a practical use in the third one, the use, in order to solve recurring problems. In all three phases described so far, the design pattern can be continuously evaluated.

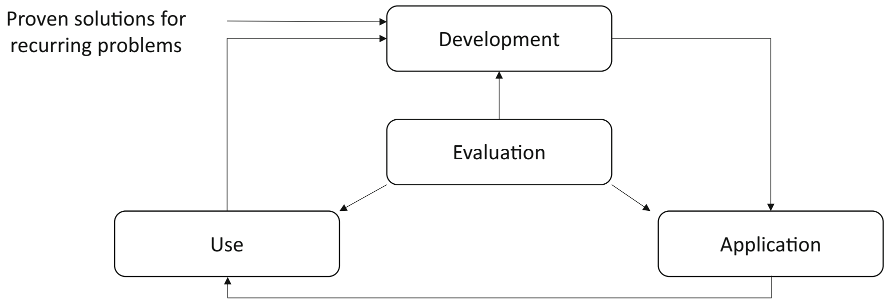


Fig. 2 Design pattern life cycle (based on [32])

5 Problem-Space and Solution-Space in System Development

DSR projects aim to solve problems in application domains. In doing so, a detailed understanding and description of the problem, as well as the positioning of those within the problem-space, are essential. Two key components describe the problem-space: the application context and the quality criteria for solution acceptance [5].

The application context describes the problem in its context. The problem-space, the main stakeholders of the problem-space, the mutual influence of those with the design solution are discussed. In addition, problem-spaces are closely tied to time and place. The relevance of today's problem may not be tomorrow [5]. Therefore, it is essential to record the time in which the problem was perceived. Contextual aspects of location include relevant geographic indications such as rural vs. urban environments or already developed vs. developing countries. Overall, the context of the application of a DSR project defines an ideographic basis for the dissemination of design knowledge.

The second key component deals with how well the design solves the problem. In describing the quality criteria for the problem, the sociotechnical aspects of the design solution must be recognized. Therefore, design requirements include mixed objectives from technology, quality information, human interaction, and societal needs. This is accompanied by acceptance criteria for evaluating potential design solutions and guides designing both formative and summative evaluation methods.

Thus, positioning a DSR project in the problem-space establishes the situational context and research goals of the project (i.e., quality criteria for design innovation) [33]. The effective reuse of design knowledge for future research depends on how well this problem-space can be transferred to the new research projects. The transferability of design knowledge provides information on how well new research contexts and objectives adapt to the knowledge base. This context can be described in terms of several dimensions, including domain, stakeholder, time, and place. Low transferability of design knowledge in a project indicates a very specific context with restrictive goals. In contrast, high transferability of design knowledge and more general applications of it to problem domains would support—even within and/or between different application domains. Legner et al. [29] found that knowledge accumulation occurred incrementally as a result of maturing abstract and situational domain knowledge (solution-space) and in response to evolving roles of data (problem-space).

The design knowledge in solution-space comprises the knowledge for solving related problems. It includes not only results but also activities of the DSR. Results of DSR can take different forms, such as artifacts as well as design patterns or theories. Artifacts should support replication and reuse by future research projects. Design theories and principles help understand how and why artifacts meet the goals of the problem-space.

However, knowledge in the solution-space may also relate to design processes and the evaluation of activities to design, evaluate, and refine DSR results in iterative cycles. Design activities include a search process to identify the best design candidates from the solution-space. Information about goodness criteria from the problem-space is used for value maximization while constrained by resource availability and applicability. For the reusability of design knowledge in the future, it is important to refer to the fundamentals of design. This can be done through kernel theories as well as by recording creative insights that have led to innovative design improvements.

In the solution-space, the suitability of solutions varies depending on the selected target problem. Design activities within a project can cover a larger part of the problem-space than solutions from research. The more suitable a solution is, the more operational it is for the users in the application. The degree of suitability refers to the normative solution power. More unsuitable solutions may have a lower normative force to guide problem-solving behavior. This makes the solution less prescriptive compared to a manual. Thus, the lower the fitness of the solution, the greater the effort required to apply the design knowledge to a new problem.

There is a trade-off between transferability and suitability of design knowledge. Often, higher suitability implies more restriction to a particular context. In turn, a less fit representation of design knowledge may support higher transferability. Techniques for representing design knowledge that enables reusability need to be developed. Exemplary of this is configurational configuration models or methods that allow the tradeoff between the transferability and suitability of design knowledge to be managed [5]. Using such techniques, design knowledge is represented for alternative solution variants that fit in different contexts of the problem-space.

6 Design Patterns to Close Knowledge Gaps

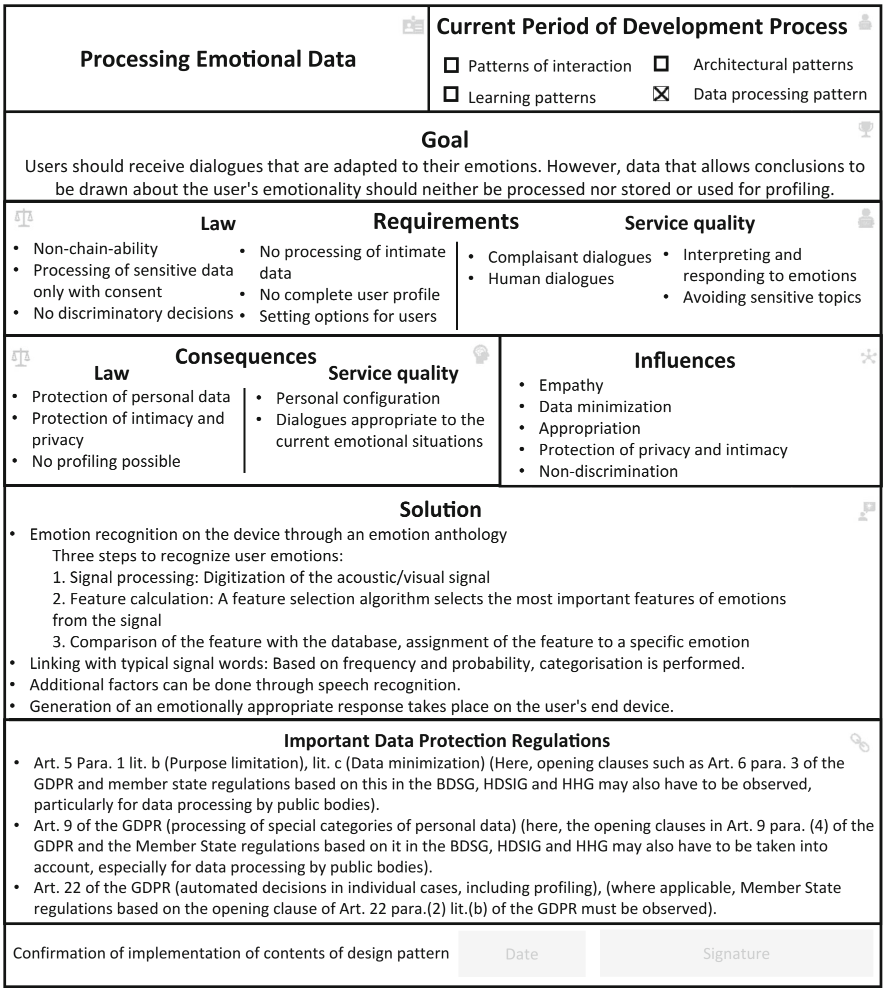
Developers of novel technologies, such as AI-based smart personal assistants [34], acquire design knowledge through experience, the training they have received, and problems they have already solved, which they can use to solve problems that arise in the future. Here, the amount of existing design knowledge in the person's problem/solution-space differs depending on the person's wealth of experience and the training they have received so far. According to Nonaka [8], the existing design knowledge of a developer can be called tacit knowledge. This must be made available to others via externalization first and can be done either orally or in writing through codification.

Design patterns externalize tacit design knowledge by codifying it for other people and recording it in written form. With the help of the externalized design knowledge, (design) problems can now be solved. While experienced developers have access to a wealth of experience, inexperienced developers usually have significantly fewer “best practices”. As a result, the solution-space also differs depending on the level of experience. Due to their abstract representation, design patterns do not offer a so-called recipe for problems to be solved but support the user to find a suitable solution for the problem at hand by pointing out possible solution approaches. Thus, a design pattern can be the solution for many problems.

The user of the design pattern is in the problem-space with the problem at hand and searches for a suitable solution in the solution-space. It should be noted that for numerous problems there is no single correct solution, but several solutions can solve the problem satisfactorily. The design pattern provides the user with food for thought about possible solutions, which expands the user’s solution-space and allows him to search individually for a suitable solution.

According to vom Brocke et al. [5], patterns are a component of the design knowledge base. They help to find suitable solutions for existing problems. By providing information about the problem context, it helps to understand the problem to be solved in detail and thus to find the truly appropriate solution. In addition to the description of the problem, a core component of a pattern is the solution and an associated description of the procedure for solving the problem. The elements described so far contain information directly from the problem or solution-space. However, the user is missing information that describes statements about effects of the implementation of the design pattern on the technology or helps him to select the right pattern. These elements are located between the problem-space and the solution-space and can be categorized as evaluation according to vom Brocke [5]. The user receives knowledge from the design pattern, which describes why the practical implementation of the pattern is necessary. A description of the target state after the successful implementation of the pattern supports the user to select and understand the pattern.

A design pattern not only presents the solution, it also describes the problem to be solved. Depending on the domain and application context, the pattern refers to the requirements to be solved or the problem to be solved. As a result, a pattern encompasses aspects of both the problem-space and the solution-space and links the two domains into each other. This allows the user to draw a bridge between the problem to be solved and the possible solution. While other approaches to knowledge codification, such as wikis, cheat sheets, and design principles focus primarily on the solution to the problem, design patterns bridge the two spaces. Additional information such as user stories, consequences, or use cases supports users to generate knowledge around the problem domain and thus changes the way they search for solutions.



Processing Emotional Data

Current Period of Development Process

☐ Patterns of interaction ☐ Architectural patterns
☐ Learning patterns [✓ Data processing pattern]

Goal

Users should receive dialogues that are adapted to their emotions. However, data that allows conclusions to be drawn about the user's emotionality should neither be processed nor stored or used for profiling.

Requirements

Law
- Non-chain-ability
- Processing of sensitive data only with consent
- No discriminatory decisions
- No processing of intimate data
- No complete user profile
- Setting options for users

Service quality
- Complaisant dialogues
- Human dialogues
- Interpreting and responding to emotions
- Avoiding sensitive topics

Consequences

Law
- Protection of personal data
- Protection of intimacy and privacy
- No profiling possible

Service quality
- Personal configuration
- Dialogues appropriate to the current emotional situations

Influences
- Empathy
- Data minimization
- Appropriation
- Protection of privacy and intimacy
- Non-discrimination

Solution
- Emotion recognition on the device through an emotion anthology
  Three steps to recognize user emotions:
  1. Signal processing: Digitization of the acoustic/visual signal
  2. Feature calculation: A feature selection algorithm selects the most important features of emotions from the signal
  3. Comparison of the feature with the database, assignment of the feature to a specific emotion
- Linking with typical signal words: Based on frequency and probability, categorisation is performed.
- Additional factors can be done through speech recognition.
- Generation of an emotionally appropriate response takes place on the user's end device.

Important Data Protection Regulations
- Art. 5 Para. 1 lit. b (Purpose limitation), lit. c (Data minimization) (Here, opening clauses such as Art. 6 para. 3 of the GDPR and member state regulations based on this in the BDSG, HDSIG and HHG may also have to be observed, particularly for data processing by public bodies).
- Art. 9 of the GDPR (processing of special categories of personal data) (here, the opening clauses in Art. 9 para. (4) of the GDPR and the Member State regulations based on it in the BDSG, HDSIG and HHG may also have to be taken into account, especially for data processing by public bodies).
- Art. 22 of the GDPR (automated decisions in individual cases, including profiling), (where applicable, Member State regulations based on the opening clause of Art. 22 para.(2) lit.(b) of the GDPR must be observed).

Confirmation of implementation of contents of design pattern Date Signature

Fig. 3 Design pattern "Processing Emotional Data"

7 Design Patterns and Their Relationship to Design Principles

In IS research, design principles have been established in recent years as a proven means of codifying design knowledge [35, 36]. Design principles provide users with necessary information about the design of technologies in the shortest possible time. In the DSR paradigm, design principles are often formulated together with design requirements and represent concrete specifications for the design of the technology.

The work from Kruse et al. [36] classified design principles into three categories: a) action-oriented, b) materiality-oriented, and c) action and materiality-oriented. Design patterns can also be classified into these categories [6], but they contain further information that goes beyond these categories. This is because design knowledge is generated and tested through the application and use of design patterns. The codified design knowledge in design patterns goes beyond a mere instruction to do something and offers an explanation of how to implement it. Design patterns act as a bridge between design knowledge and the developed technology. Compared to design principles, design patterns additionally present action-oriented and material information. And exactly this content differs design patterns from design principles. The purpose of design principles can be described as a rule or behavioral standard [37] that gives precise instructions on how an artifact must be built [38]. Design patterns, on the other hand, contain information that conveys to the user the intention of why something must be implemented.

Figure 3 shows the exemplary design pattern "Processing Emotional Data". The design pattern contains elements such as target state, problem, effects, and solution, which provide the user with extensive information. With the help of the abstract solution approach in the pattern, a solution is searched for in the user's solution space for the problem that exists in practice. Another crucial feature is that design patterns gain added value with increasing time and application. By repeatedly establishing the solution presented, it is sharpened and must prove itself in a large number of projects. Design principles, on the other hand, are usually developed with a strong technology focus and are not updated over time.

8 Conclusion

In this paper, we show the specifics of design knowledge and the benefits of codifying design knowledge. Especially, in case of practical useabal design solutions approaches to codifing the knowledge may be needed. Thus, we demonstrate how design knowledge is externalized through design patterns by using the SECI model and Nonaka's thoughts [1]. Our contribution is intended to contrast design patterns with other approaches for codifying knowledge and work out its practical applicability. Thus, we propose that design patterns may close knowledge gaps and provide fruitful interactions between the problem-space and the solution-space.

References

1. Gregor, S., Jones, D.: The anatomy of a design theory. Assoc. Inf. Syst. (2007)
2. Österle, H., Becker, J., Frank, U., Hess, T., Karagiannis, D., Krcmar, H., Loos, P., Mertens, P., Oberweis, A., Sinz, E.J.: Memorandum on design-oriented information systems research. Eur. J. Inf. Syst. **20**, 7–10 (2011)

3. Peffers, K., Tuunanen, T., Niehaves, B.: Design science research genres: introduction to the special issue on exemplars and criteria for applicable design science research. Eur. J. Inf. Syst. **27**, 129–139 (2018)
4. Nonaka, I., Toyama, R.: The knowledge-creating theory revisited: knowledge creation as a synthesizing process. Knowl. Manag. Res. Pract. **1**, 2–10 (2003)
5. vom Brocke, J., Winter, R., Hevner, A., Maedche, A.: Accumulation and evolution of design knowledge in design science research–A journey through time and space. JAIS. (2019)
6. Dickhaut, E., Li, M.M., Janson, A.: Developing lawful technologies – a revelatory case study on design patterns. HICSS. 54 (2021 in Erscheinung)
7. Müller, R.M., Thoring, K.: A typology of design knowledge: a theoretical framework. AMCIS. (2010)
8. Nonaka, I., Takeuchi, H.: The Knowledge-Creating Company. How Japanese Companies Create the Dynamics of Innovation. Oxford University Press (1995)
9. Faraj, S., Sproull, L.: Coordinating expertise in software development teams. Manag. Sci. 1554–1568 (2000)
10. Lukyanenko, R., Jeffrey, P.: Design Theory Indeterminacy: What is it, how can it be reduced, and why did the polar bear drown? JAIS. 1–59 (2020)
11. Garud, R.: On the distinction between know-how, know-what, and know-why. Adv. Strat. Manag. 81–102 (1997)
12. Baskerville, R., Baiyere, A., Gergor, S., Hevner, A., Rossi, M.: Design science research contributions: finding a balance between artifact and theory. JAIS. **19**, 358–376 (2018)
13. Baxter, G., Sommerville, I.: Socio-technical systems: from design methods to systems engineering. Interact. Comput. **23**, 4–17 (2011)
14. Zhang, G., Su, X., Zhou, G.: Blog-based dynamic organization and management for multidiscipline knowledge. 515–517 (2010)
15. Morse, W.C., Nielsen-Pincus, M., Force, J.E., Wulfhorst, J.D.: Bridges and barriers to developing and conducting interdisciplinary graduate-student team research. Ecol. Soc. **12**(2), 8 (2007)
16. Dickhaut, E., Janson, A., Roßnagel, A., Leimeister, J.M.: Interdisziplinäre Anforderungsmuster für smarte persönliche Assistenten. Mittel zu Erfassung divergenter Anforderungen aus Informatik und Recht. Datenschutz und Datensicherheit (DuD) (2020)
17. Dickhaut, E., Thies, L.F., Janson, A.: Die Kodifizierung von Gestaltungswissen in interdisziplinären Entwurfsmustern. Lösungen im Spannungsfeld zwischen Rechtsverträglichkeit und Dienstleistungsqualität. Datenschutz und Datensicherheit (DuD) (2020)
18. Zenkert, J., Holland, A., Fathi, M.: Discovering contextual knowledge with associated information in dimensional structured knowledge bases. International Conference on Systems, Man, and Cybernetics, pp. 923–928 (2016)
19. Wheeldon, J., Faubert, J.: Framing experience: concept maps, mind maps, and data collection in qualitative research. Int J Qual Methods. **8**, 68–83 (2009)
20. Phuwanartnurak, A.J.: Interdisciplinary Collaboration through Wikis in Software Development. IEEE, Piscataway, NJ (2009)
21. Winkler, M., Huber, T., Dibbern, J.: The Software Prototype as Digital Boundary Object: A Revelatory Longitudinal Innovation Case. (2014)
22. Seidel, S., Chandra Kruse, L., Székely, N., Gau, M., Stieger, D.: Design principles for sensemaking support systems in environmental sustainability transformations. Eur. J. Inf. Syst. **27**, 221–247 (2018)
23. Alexander, C.: The timeless way of building. Oxford University Press, New York (1979)
24. Song, Y., Guo, Y., Thuente, D.: A quantitative case study on students' strategy for using authorized cheat-sheets. IEEE Frontiers in Education, pp. 1–9 (2016)
25. Hsiao, I.-H., Lopez, C.: Lessons learned from students' cheat sheets: generic models for designing programming study guides. IEEE 16th International Conference on Advanced Learning Technologies (ICALT), pp. 209–211 (2016)

26. Kaindl, H., Falb, J., Melbinger, S., Bruckmayer, T.: An approach to method-tool coupling for software development. Fifth International Conference on Software Engineering Advances, pp. 101–106 (2010)
27. Pawlowski, R.: Bridging user organizations: knowledge brokering and the work of information technology professionals. MIS Q. **28**, 645 (2004)
28. Winkler, M., Brown, C., Huber, T.L.: Recurrent knowledge boundaries in outsourced software projects: a longitudinal study. ECIS. (2015)
29. Legner, C., Pentek, T., Otto, B.: Accumulating design knowledge with reference models: insights from 12 years' research into data management. J. Assoc. Inf. Syst. (JAIS). (2020)
30. Alexander, C.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press (1977)
31. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object Oriented Software. Addison Wesley Professional (1994)
32. Petter, S., Khazanchi, D., Murphy, J.D.: A design science based evaluation framework for patterns. DESRIST (2008)
33. Gregor, S., Hevner, A.R.: Positioning and presenting design science research for maximum impact. MIS Q. 337–355 (2013)
34. Knote, R., Janson, A., Söllner, M., Leimeister, J.M.: Value co-creation in smart services: a functional affordances perspective on smart personal assistants. J. Assoc. Inf. Syst. (JAIS). (2020)
35. Chandra Kruse, L., Nickerson, J.V.: Portraying Design Essence. 51st Hawaii International Conference (2018)
36. Seidel, S., Chandra Kruse, L., Székely, N., Gau, M., Stieger, D.: Design principles for sensemaking support systems in environmental sustainability transformations. EJIS. **27**, 221–247 (2018)
37. Bellucci, E., Zeleznikow, J.: Managing Negotiation Knowledge with the goal of developing Negotiation Decision Support Systems. ACIS 2005 Proceedings (2005)
38. Chandra, L., Seidel, S., Gregor, S.: Prescriptive knowledge in IS research: conceptualizing design principles in terms of materiality, action, and boundary conditions. HICSS. 4039–4048 (2014)