Human Systems Engineering and Design (IHSED 2026), Vol. 237, 2026, 23–33
https://doi.org/10.54941/ahfe1008142

Development of a Design System for an AI Call Intake Assistant—A User-Centered Reverse Engineering Approach

Joshua Kopp[1], Stefan Pfeffer[1], Lukas Willmann[2], Thomas Schlegel[2], and Verena Wagner-Hartl[1]

[1]Faculty Engineering & Technology, Campus Tuttlingen, Furtwangen University, Tuttlingen, 78532, Germany
[2]Institute for Intelligent Interactive Ubiquitous Systems, Furtwangen University, Furtwangen, 78120, Germany

ABSTRACT

Although user-centered design (UCD) advocates for early user involvement, practical constraints often result in Human Factors integration taking place only when prototypes already exist. This paper proposes a solution to this challenge by employing a "reverse-engineered" UCD approach to an existing AI call intake assistant prototype. Deviating from the traditional linear approach, this approach reconstructs the problem space directly from the solution space. To apply this approach, a multi-stage method was utilized. First, a group interview was conducted with five developers of the existing AI prototype to elicit their implicit design rationales and assumptions. From this interview, preliminary usage requirements were retrospectively derived. Secondly, these requirements were validated and refined through a focus group involving 14 dispatchers, followed by a Kano analysis to identify basic, performance, and excitement factors. The study revealed critical discrepancies between developer assumptions and actual user needs. For instance, dispatchers required the capacity to intervene and cancel the AI assistance at any moment. The consolidated set of requirements now serves as the foundation for the next development phase of the AI assistant, ensuring better alignment with actual work practices. This demonstrates how a prototype can serve as a probe to deliberately step back into the problem space. Furthermore, the reverse-engineered approach holds particular promise for future rapid AI-based prototyping, especially when user involvement occurs late. This ensures the continued efficacy of human-centered design principles.

Keywords: User-centered design process, Reverse engineering approach, AI call intake assistant

INTRODUCTION

User-centered design (UCD) emphasizes early and continuous involvement of users to ensure that interactive systems fit their needs and work practices (DIN EN ISO 9241-210, 2019). In practice, however, users and human factors (HF) specialists are often involved only late in the process, typically when an initial prototype already exists and key architectural decisions have been taken. In the domain of safety-critical systems, such as dispatcher tools or control room interfaces, this late integration can be exacerbated by increasing

Received June 11, 2026; Revised July 2, 2026; Accepted July 23, 2026; Available online September 25, 2026

technical complexity and stringent safety regulations (e.g., restricting user involvement to mandatory verification and validation testing at the end of the development cycle; Koskinen et al., 2022). Furthermore, because these highly specialized internal systems are rarely accessible to the public, their specific interaction patterns remain underrepresented in current HCI literature (e.g., Koskinen et al., 2022; Bachmann et al., 2022). Consequently, conformity with user expectations and fundamental interaction principles (DIN EN ISO 9241-110, 2020) often emerges implicitly, driven by technical evolution rather than early user-centered design. This can create a significant risk: traditional UCD methods alone may be insufficient to capture the full complexity and potential hazards of modern safety-critical devices, necessitating a deeper, more formal understanding of the underlying system logic (Thimbleby, 2007). This challenge is especially pronounced for complex AI-based assistants, where the "black box" nature of the technology adds lack of transparency. Here, a reverse engineering approach can prove particularly valuable, as it enables researchers to systematically extract expert heuristics and "built-in" operational knowledge directly from these untransparent existing systems.

User-centered design processes are conventionally illustrated by the Double Diamond model (Design Council, 2005), which prescribes an initial divergent–convergent cycle for problem framing (discover/define) — the first diamond — followed by a second cycle for solution development (develop/deliver) — the second diamond. When HF experts enter the process at a point where a prototype already exists, they find themselves effectively mid-way through the second diamond: the solution development phase is already underway, yet there is typically no systematic record of the design rationales, assumptions, and implicit requirements that shaped the prototype. The prototype itself, however, embodies rich domain knowledge and work-specific interaction patterns.

From the authors' perspective, several reasons can be identified as to why the problem space is often inadequately explored or skipped entirely in practice. These reasons are frequently discussed in UCD literature (Norman, 2013; Gothelf and Seiden, 2021). In product development, concrete solutions such as features or prototypes are primarily considered as relevant and visible outcomes, while a deep understanding of the problem space is rarely recognized as a standalone output. Since organizational progress is predominantly evaluated based on such visible and tangible artifacts, engaging with the problem space often appears unproductive or even like a standstill.

Furthermore, established planning and management frameworks in projects require early commitments, which may structurally disadvantage open exploratory phases. The problem space is frequently characterized by so-called "unknown unknowns". In other words, these are aspects whose existence and relevance are initially unknown. This inherent indeterminacy largely eludes classical planning and makes it difficult to precisely define goals, efforts, or outcomes in advance. Unfortunately, an insufficiently defined problem space contrasts with organizational requirements for predictability and decision-making, as it introduces uncertainty into planning and can delay critical decisions. At the same time, every blind spot in the problem space poses a tangible risk to development.

Furthermore, many departments have traditionally been strongly solution-oriented: particularly in engineering and management contexts, the prevailing approach focuses on rapid problem-solving, while the systematic and in-depth definition of problems is less firmly established as a distinct skill and practice. To systematically reconstruct this embedded but undocumented knowledge, HF specialists can perform a form of "Design Recovery" (Chikofsky and Cross, 1990). While the term reverse engineering is traditionally associated with software code, its application to user interfaces is a recognized field in Human-Computer Interaction (Montero et al., 2010; 2013). Applied to a UCD context, the prototype ceases to be a final product and instead becomes a primary source for an inquiry process. However, to ensure that these reverse-engineered requirements do not merely reflect developer bias, a formal multi stage validation process involving expert judgments and end user feedback is essential (Quiñones et al., 2018).

To address this methodological challenge, the present paper explores the application of a reverse engineered UCD process using an existing AI-based call intake assistant for dispatcher-driver communication as a case study. In this special field, efficient communication between dispatchers and drivers is especially important to maintain a smooth operation in public transport (Bachmann et al., 2022). Therefore, in public transport control rooms, dispatchers close the feedback loop of daily operations in real time, a safety-relevant task whose complexity increase with e.g., denser networks, multimodal services, and KRITIS-level reliability demands (Schnieder, 2018). Here, modern AI-based methods, predictive disposition, anomaly detection, and learning decision support may reveal design possibilities to support the operator during routine tasks so that cognitive resources remain available for any extraordinary events that may occur. Thus, an AI-assist is used as sample application for the presented case study. The primary objective is to systematically reconstruct user needs directly from the solution space, taking a deliberate step back into the problem space. Rather than bypassing problem definition entirely, the presented study utilizes the existing prototype as an active inquiry probe. The aim of the used approach is to derive previously implicit requirements, identify potential mismatches between developer assumptions and actual dispatcher needs, and in particular, to provide a structured framework for integrating human factors even when user involvement occurs late in the development cycle.

METHOD

To achieve the outlined objectives and address the theoretical and practical challenges of integrating UCD into concept stage, this approach employed a multi-stage, qualitative research design centered around an existing AI call intake assistant prototype. The methodological framework adapts the conventional user-centered design (UCD) approach (see DIN EN ISO 9241-210, 2019) to fit the development's "solution-first" nature. By developing and applying a reverse-engineered UCD process, the aim was to systematically extract, validate, and prioritize system requirements retrospectively.

Before describing the empirical procedure, it is essential to understand the technical specifications of the AI call intake prototype, which serves as example

for the presented approach. The prototype implements a communication-centered analysis pipeline for dispatcher control stations, organized along a capture (classification) adaptation chain. Incoming voice traffic is acquired and transcribed by a speech-to-text application. The resulting utterances are forwarded to a multi-classifier ensemble that combines rule-based, classical machine-learning, and LLM-based components and is exposed via a REST interface; the ensemble assigns each message a domain-specific category together with a confidence vector. These results are propagated through an event-driven backend, which manages public-transport entities such as vehicles, journeys, stops, and routes. In its future form, the system will thus cover the capture, analysis, and classification of operator communication.

As presented before, to systematically extract requirements from this existing prototype, a reverse-engineered UCD approach was developed. While the standard Double Diamond model (see Figure 1) begins with a forward discover/define cycle before any concrete solution exists, this process fundamentally shifts this dynamic (Design Council, 2005).

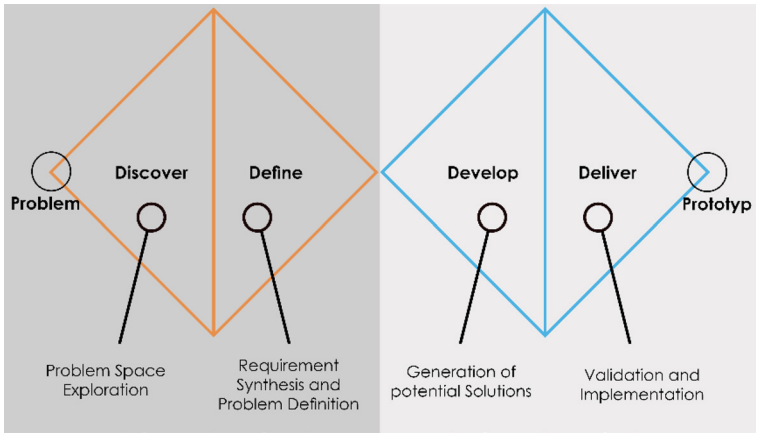

Figure 1: The classic double diamond model illustrating a traditional, forward moving UCD process (adopted from Design Council, 2005).

Because the development of the AI call intake assistant started in the solution space, the reverse-engineered process inserts a reverse discover–define and specifying loop (see Figure 2). This process reconstructs the tacit knowledge of developers and human factors experts and uses it to derive requirements.

The reverse-engineered UCD process consists of four interconnected phases. Starting from an existing prototype, Phase 1 (Reverse Discover) reconstructs the developers' implicit knowledge and design rationales. Phase 2 (Define and specifying) synthesizes this knowledge with the expertise of a Human Factors specialist to derive preliminary usage requirements. Phase 3 (Diversify) validates and refines these requirements with end users (dispatchers), and Phase 4 (Deliver) applies a Kano analysis (Kano et al., 1984) to prioritize them for further development.

To execute the first phase (Reverse Discover), five developers participated in a 30-minute semi-structured group interview. The interview covered five thematic blocks: (1) system overview and goals, (2) implemented functions, (3) design decisions, (4) technical constraints, and (5) planned extensions.

For each module (stop search, stop details, map, individual traffic, and notifications), developers described use cases, rationales, and limitations (e.g., handling information density or duplication between search and map). Notes were synthesized to systematically document the developers' underlying design rationales and operational assumptions.

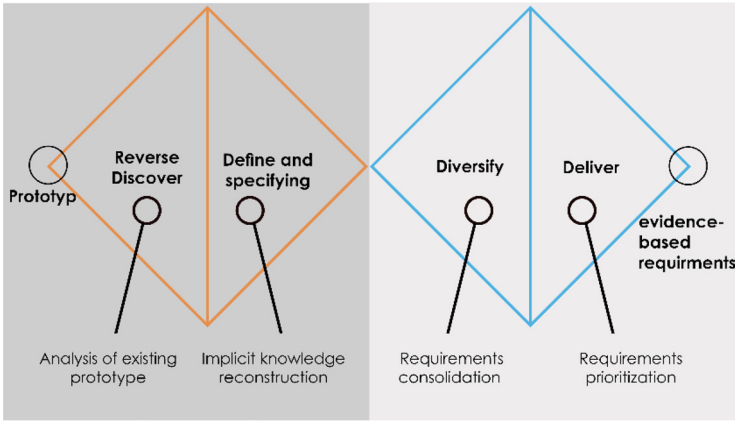

Figure 2: Overview of the proposed reverse-engineered UCD process, consisting of four interconnected phases starting from the prototype (adopted from Design Council, 2005).

In the second phase (Define and specifying; see also Figure 2), the documented assumptions of the developers were combined with the domain knowledge of a Human Factors expert intimately familiar with the dispatchers' operational context. This synthesis translated the implicit technical knowledge into 15 preliminary usage requirements, ensuring they reflected both the developers' assumptions and practical workplace constraints.

Following this, Phase 3 (Diversify) was conducted via a focus group with 14 dispatchers. They reviewed the prototype alongside the preliminary usage requirements, assessing their validity, identifying mismatches, and suggesting refinements. Transcripts were analyzed using qualitative content analysis (based on Mayring, 2014). This collaborative validation added three new usage requirements in addition to the 15 identified in Phase 2 (particularly concerning the AI-Hotline), resulting in a refined set of 18 usage requirements. In the fourth and final phase (Deliver), the 18 usage requirements were prioritized using a Kano analyses (Kano et al., 1984), completed by the dispatchers and focusing on AI-Hotline features (e.g., "voicebot with adaptive dialogue" or "fail safe: direct transfer to dispatcher"). Each feature was rated in both its functional and dysfunctional form using a 5-point scale ("I like it," "I expect it," "I am neutral," "I can tolerate it," and "I dislike it"). The paired responses for each feature were then cross-tabulated using the standard Kano evaluation matrix and classified as must-be, performance, attractive, or indifferent requirements (Kano et al., 1984).

RESULTS

The application of the reverse-engineered UCD process yielded key insights into the significant discrepancies between the initial technical assumptions

and actual user needs regarding the AI call intake assistant. These findings are presented in accordance with the newly structured four-phase methodology, with the initial discovery and definition phases grouped together for clarity.

Phases 1 and 2 (Reverse Discover and Define and specifying) focused on extracting developer rationales and synthesizing them into preliminary requirements. The data analysis of the developer interviews, combined with the human factors expert review, revealed that the 15 preliminary usage requirements were characterized by a predominantly technology-driven focus. The findings indicate that the developers operated under the implicit assumption that the AI should aim for maximum automation to reduce the dispatchers' cognitive load and waiting times. For instance, the results show that the initial system design prioritized an autonomous voicebot designed to capture and structure core data independently before any human intervention occurs. Furthermore, there was an underlying assumption that the AI could autonomously categorize and prioritize the dispatchers' call queue based on technical urgency parameters.

Within Phase 3 (Diversify), these preliminary requirements were presented to 14 dispatchers during a focus group session. This stage revealed significant mismatches: the qualitative analysis of the transcripts showed that dispatchers perceived the highly automated, "black box" approach not as helpful assistance, but as a severe loss of operational control. The end users emphasized the unpredictable nature of their work and strongly objected to the AI acting as an unavoidable barrier between them and the callers. Instead, they formulated additional requirements centered on transparency and human-in-the-loop capabilities. Table 1 illustrates these core discrepancies between the initial technical assumptions and the dispatchers' operational realities across selected system modules.

Table 1: Key mismatches between initial developer assumptions (Phases 1–2) and actual dispatcher requirements (Phase 3) across selected system modules.

System Module	Developer Assumption (Phase 1 & 2)	Dispatcher Requirement (Phase 3)
General UI / Map	The system automatically hides less relevant information during a high density of error messages to reduce cognitive load.	The dispatcher must manually control layer transparency and information density; the system must not hide data autonomously.
AI-Hotline	The voicebot autonomously filters and structures caller data before the dispatcher is involved.	The dispatcher needs a prominent, always available "fail safe" to immediately bypass the AI and talk to the driver directly if needed.
Private traffic	An algorithm automatically creates and executes rerouting suggestions based on real time traffic jams.	Rerouting algorithms must act purely as advisory tools; the final execution requires explicit manual confirmation of the dispatcher.

Phase 4 (Deliver) concluded the process by evaluating the dispatcher responses using a Kano analysis to systematically prioritize the refined requirements. The results of this classification are summarized in Table 2, which contrasts selected features with their process origin and their resulting Kano category. Notably, requirements newly identified during the focus group (Phase 3) were assessed as the most critical. Specifically, the "fail-safe" function (REQ 01) and the absolute capability to interrupt the AI (REQ 02) were classified as must-be or performance requirements. As shown in Table 2, the absence of these control-oriented features would lead to an absolute rejection of the system. In contrast, features originally prioritized during the initial technical phases (Phases 1 & 2), such as the "voicebot with adaptive dialogue" (REQ 03) and "AI-based prioritization" (REQ 04), were primarily categorized as attractive factors. This highlights a clear disconnect: while the initial technological perspective assumed that automation was the primary benefit, users prioritize override capabilities in critical situations. Ultimately, this empirically grounded classification provides a prioritized roadmap anchored in dispatcher reality.

Table 2: Kano classification of selected AI assistant system requirements based on dispatcher evaluations ($N = 14$).

REQ. ID	AI Assistant Feature	Origin (Phase)	Kano Category
REQ 01	Fail Safe: Direct transfer to dispatcher	Phase 3 (focus group)	Must-be / Performance
REQ 02	Absolute capability to interrupt the AI	Phase 3 (focus group)	Performance
REQ 03	Voicebot with adaptive dialogue	Phases 1 & 2 (technical assumptions)	Attractive (excitement)
REQ 04	AI-based prioritization of incoming calls	Phases 1 & 2 (technical assumptions)	Attractive (excitement)
REQ 05	Structured capture of core data (location, etc.)	Phases 1 & 2 (technical assumptions)	Indifferent (neutral)
REQ 06	Structured forwarding of data to UI	Phases 1 & 2 (technical assumptions)	Indifferent (neutral)

DISCUSSION

As mentioned before, the findings illustrate the practical utility of the presented reverse-engineered User-Centered Design approach, particularly when human factors integration occurs late in the development of complex systems, as demonstrated by the case study of an Artificial Intelligence (AI) call intake assistant for public transport dispatching. While traditional human-centered design frameworks prioritize extensive upfront problem exploration, industrial realities often necessitate a solution-first development approach due to prevailing project constraints. When human factors experts are involved late in such a process, the existing prototype is frequently

treated solely as an artifact requiring usability corrections. However, this paper demonstrates that treating the prototype as a tangible analytical object rather than solely as a usability artefact, enables highly effective, retrospective requirement elicitation.

As demonstrated by the presented case study, the reversed engineering approach made the developers' underlying assumptions transparent by systematically uncovering the reasoning behind the prototype. Furthermore, and even more importantly, presenting these derived concrete design choices to the dispatchers resulted in highly specific feedback that would probably have been missed in abstract, early-stage interviews. This observation anchors practical results directly within established theoretical discourse. Moreover, as outlined by Thimbleby (2007) and Koskinen et al. (2022), eliciting requirements in safety-critical domains presents a fundamental challenge: without a tangible reference, users often find it difficult to imagine and articulate their needs regarding abstract AI behaviors or rare emergency situations. In this context, the prototype can serve as a concrete reference point that allowed dispatchers to evaluate the practical implications of the developers' automation logic. This direct experience enabled the dispatchers to explicitly articulate the necessity for human-in-the-loop capabilities as a fundamental requirement. This result is in line with well-established human factors principles regarding different levels of automation in complex systems (cf. Parasuraman et al., 2000).

Based on these qualitative insights, a key objective was to systematically evaluate the actual weight of the users' demands. Here, the Kano analysis (Kano et al., 1984) was used as a structured method to quantify the previously subjective dispatcher feedback. By classifying the manual fail-safe and transparency features predominantly as "must-be" requirements, the analysis provided strong empirical indications that AI autonomy should not override human operational control in this specific dispatching context. While these findings strongly suggest that preserving human agency is critical, they represent an initial evaluation that requires further validation in broader, live operational settings. Nevertheless, this systematic categorization effectively corrected the developers' initial assumption that technical capability alone equates to operational desirability.

Consequently, while the presented approach successfully realigned the prototype with actual user needs, the interpretation of these findings must be considered in the light of several methodological boundaries. First, the evaluation was applied to a single AI assistant within a highly specialized dispatching context, involving a relatively small qualitative sample size ($N = 14$). While sufficient for uncovering rich qualitative insights — consistent with established thresholds for theoretical saturation (Guest et al., 2006) — this sample size limits statistical generalizability and necessitates broader quantitative verification in future deployments. Furthermore, while the methodological framework is inherently transferable to other safety-critical systems and complex software domains, specific user requirements remain highly domain-dependent, particularly regarding requirements for fail-safe mechanisms. Finally, the retrospective extraction of design rationales relies

heavily on the developers' ability to recall and articulate their decision-making processes accurately, which may be subject to post-rationalization biases inherent to retrospective recall. To address these limitations and empirically validate the broader applicability of the proposed approach, further research and subsequent case studies across different domains are required.

CONCLUSION

In an ideal development environment, user-centered design (UCD) begins long before the first line of code will be written. In reality, however, technical feasibility, rapid prototyping, and organizational constraints often precede human factors integration. This paper proposed and applied a reverse-engineered UCD approach to bridge the gap between an existing AI call intake assistant prototype and undocumented user needs within a safety critical dispatching environment. By systematically reconstructing developer assumptions and validating them against actual dispatcher realities, a solution driven AI prototype was successfully guided back into the problem space. Consequently, the reverse-engineering approach was revealed to be highly effective in identifying a critical mismatch: while developers prioritized an autonomous, AI-driven call handling process to reduce cognitive workload, dispatchers strictly required prominent human-in-the-loop capabilities. Features such as a direct manual fail-safe and the ability to interrupt the AI assistant at any time were identified not only as preferences, but as essential requirements for maintaining operational control.

Prospectively, the Kano-prioritized set of 18 requirements will serve as the empirical foundation for the next iteration of the AI assistant, ensuring that it is experienced as a transparent tool rather than a restrictive "black box". Nevertheless, this study demonstrates that late user involvement does not exclude effective human-centered design; but, it requires a methodological shift from generative ideation to systematic, prototype-driven requirement reconstruction.

The findings of this study have a number of important implications for future practice and the presented methodology holds significant potential for broader generalization. As advanced AI tools, rapid prototyping, and emerging software engineering trends like end-user development or "vibe coding" dramatically accelerate the speed at which functional software can be built, traditional, time-intensive UCD phases risk being increasingly bypassed. In such a fast-paced development landscape, using rapid AI prototypes as active inquiry probes offers a highly pragmatic framework. It acknowledges that while early user involvement remains the gold standard, systematic requirement reconstruction serves as a necessary and effective safety net to bridge the gap between rapid technological feasibility and genuine human-centricity.

To sum up, this study demonstrates that effective human-centered design is achievable even when user involvement occurs late in the development cycle, provided a systematic methodological framework is applied. The presented reverse-engineering UCD approach enables practitioners to start from an existing prototype and work directly within the solution space, while deliberately reconstructing user needs and stepping back into

the problem space. Rather than bypassing problem definition entirely, the prototype serves as an active probe to elicit and structure requirements that were previously implicit or undocumented. Beyond the specific case study, this approach offers a transferable framework for bridging the gap between rapid technological development and genuine human-centric. This gap that is expected to intensify as AI-based tools accelerate the generation of functional prototypes across domains. Within the presented case study, the approach successfully revealed critical mismatches between developer assumptions and actual dispatcher needs, directly informing the next iteration of the AI call assistant toward better alignment with real-world work practices. The method appears particularly promising wherever prototypes emerge faster than traditional UCD phases can follow.

ACKNOWLEDGMENT

This work was funded by the Federal Ministry of Education and Research (BMBF) as part of the project IADAPT [FH-Kooperativ 1-2023: Intelligente User Interfaces für adaptive Leitstand-Systeme im öffentlichen Personenverkehr (IADAPT); 13FH056KA2]. We would like to thank all dispatchers and developers for their time and willingness to participate.

REFERENCES

Bachmann, Frank R., Briem, Lars, Busch, Florian, Vortisch, Peter. (2022). Dynamics and processes in operations control centers in urban public transport: Potentials for improvement. IEEE Transactions on Intelligent Transportation Systems, Volume 23, No. 10, pp. 17819–17834.

Chikofsky, Elliot J., Cross, James H. (1990). Reverse engineering and design recovery: A taxonomy. IEEE Software, Volume 7, No. 1, pp. 13–17.

Design Council. (2005). Eleven lessons: Managing design in eleven global brands. A study of the design process. London: Design Council.

DIN EN ISO. (2019). Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems. ISO 9241-210:2019. Geneva: International Organization for Standardization.

DIN EN ISO. (2020). Ergonomics of human-system interaction — Part 110: Interaction principles. ISO 9241-110:2020. Geneva: International Organization for Standardization.

Gothelf, Jeff, Seiden, Josh. (2021). Lean UX: Designing great products with agile teams. Boston, MA: O'Reilly Media.

Guest, Greg, Bunce, Arwen, Johnson, Laura. (2006). How many interviews are enough? Field Methods, Volume 18, No. 1, pp. 59–82.

Kano, Noriaki, Seraku, Nobuhiko, Takahashi, Fumio, Tsuji, Shin-ichi. (1984). Attractive quality and must-be quality. Journal of the Japanese Society for Quality Control, Volume 14, No. 2, pp. 39–48.

Koskinen, Hanna, Salo, Laura, Reiman, Arto. (2022). User-centered design in safety-critical environments: Challenges and methodological adaptations. Safety Science, Volume 145, p. 105495.

Mayring, Philipp. (2014). Qualitative content analysis: Theoretical foundation, basic procedures and software solution. Klagenfurt: Alpen-Adria-Universität.

Montero, Francisco, López-Jaquero, Víctor, Vanderdonckt, Jean. (2010). "Reverse engineering of user interfaces", in: Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10). New York, NY: Association for Computing Machinery, pp. 347–352.

Montero, Francisco, López-Jaquero, Víctor, Vanderdonckt, Jean. (2013). Model-driven reverse engineering of user interfaces. Information and Software Technology, Volume 55, No. 1, pp. 1–18.

Norman, Donald A. (2013). The design of everyday things: Revised and expanded edition. Philadelphia, PA: Basic Books.

Parasuraman, Raja, Sheridan, Thomas B., Wickens, Christopher D. (2000). A model for types and levels of human interaction with automation. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Volume 30, No. 3, pp. 286–297.

Quiñones, Daniela, Rusu, Cristian, Rusu, Virginica. (2018). Methodology to develop usability/user experience heuristics. Computer Standards & Interfaces, Volume 59, pp. 109–129.

Schnieder, Lars. (2018). Operational Planning in Local Public Transport: Objectives, Methods, Concepts. VDI-Buch. Berlin, Heidelberg: Springer Vieweg.

Thimbleby, Harold. (2007). Press on: Principles of interaction programming. Cambridge, MA: MIT Press.