CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

| | |
|---|---|
| Title | Knowns and unknowns: An experience report on discovering tacit knowledge of maritime surveyors |
| Authors | Sporsem, Tor;Hatling, Morten;Tkalich, Anastasiia;Stol, Klaas-Jan |
| Publication date | 2023-04-04 |
| Original Citation | Sporsem, T., Hatling, M., Tkalich, A. and Stol, K. J. (2023) 'Knowns and unknowns: An experience report on discovering tacit knowledge of maritime surveyors', in Ferrari, A. and Penzenstadler, B. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2023. Lecture Notes in Computer Science, 13975, pp. 309-323. Springer, Cham. doi: 10.1007/978-3-031-29786-1_22 |
| Type of publication | Conference item |
| Link to publisher's version | 10.1007/978-3-031-29786-1_22 |
| Rights | © 2023, the Authors, under exclusive license to Springer Nature Switzerland AG. For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. - https://creativecommons.org/licenses/by/4.0/ |
| Download date | 2026-11-03 13:26:45 |
| Item downloaded from | https://hdl.handle.net/10468/14420 |



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

# Knowns and Unknowns: An Experience Report on Discovering Tacit Knowledge of Maritime Surveyors

Tor Sporsem[1 [0000−0002−5230−7480]], Morten Hatling[1],
Anastasiia Tkalich[1 [0000−0001−7391−4194]], and Klaas-Jan Stol[1,2]

[1] SINTEF Digital, 7034 Trondheim, Norway
tor.sporsem@sintef.no
[2] University College Cork

**Abstract. [Context]** Requirements elicitation is an essential activity to ensure that systems provide the necessary functionality to users, and that they are fit for purpose. In addition to traditional 'reductionist' techniques, the use of observations and ethnography-style techniques have been proposed to identify requirements. **[Research Problem]** One frequently heard issue with observational techniques is that they are costly to use, as developers who would partake, would lose considerable development time. Observation also does not guarantee that all essential requirements are identified, and so luck plays a role. Very few experience reports exist to evaluate observational techniques in practice, and for organizations it is difficult to assess whether observation is a worthwhile activity, given its associated cost. **[Results]** This report presents experiences from DNV, a global leader providing maritime services who are renewing an information system to support its expert users. We draw on several data sources, covering insights from both developers and users. The data were collected through 9 interviews with users and developers, and over 80 hours of observation of prospective users in the maritime domain. We capture 'knowns' and 'unknowns' from both developers and users, and highlight the importance of observational studies. **[Contribution]** While observational techniques are costly to use, we conclude that essential information is uncovered, which is key for developers to understand system users and their concerns.

**Keywords:** User involvement · Expert Knowledge · Requirements engineering · Tacit Knowledge · Ethnographic Techniques

## 1 Introduction

"We can know more than we can tell"
—Polanyi [17]

"We're not good at knowing what we know"
—Ken Jennings, *Jeopardy!* champion[3] [14]

Organizations in all domains rely on software solutions to support their employees in achieving their core business goals. Automation to support professionals and experts

[3] Jennings holds the record of the longest streak of wins of the popular TV game show 'Jeopardy!'

in their manual jobs can be traced back to the early days of computing [12,21]. Initially, development processes for software were modeled after traditional engineering cycles, focusing on problem formulation and analysis, systematically developing systems based on requirements that could be identified.

Many of the early software development methods sought to provide support to systems developers in structurally and systematically design and implement systems—the so-called structured approaches; specific practices included structured analysis, structured design, data-driven design, and structured coding [5,12]. A key characteristic of these practices is that they all focus on that what can be observed and articulated; further, they take a 'reductionist' approach in that these methods consider only technical entities such as components, control structures, and data flows [5], ignoring what we could label "system-in-action" requirements that capture the subtleties of how users actually use systems.

In recognition of the importance of capturing and managing the right system requirements, the Requirements Engineering (RE) discipline emerged [15], and has been concerned with identifying, modeling, communicating, and documenting requirements [16]. Despite a very rich and mature RE literature today, RE remains a major challenge because systems are usually developed by people other than the intended users. Software systems that fail to meet the needs of expert users may threaten those users' ability to do their job and, indirectly, the core business activities of the organization. A key problem is the knowledge gap that exists between the analyst/developer, and the expert user who possesses a high level of expertise. Understanding how this gap can be closed has been a longstanding goal of the RE discipline.

As one of the opening quotes suggests, Polanyi [17] argued that much human knowledge acquired by highly skilled experts through experience is impossible to articulate. In seeking to understand whether and how expertise can be articulated, several scholars have invoked the term 'tacit knowledge' [6]. Gervasi et al. [7] drew on a notable 2002 press briefing of the late Donald Rumsfeld, Secretary of Defense during the U.S. invasion of Afghanistan and Iraq. Rumsfeld argued there are different types of knowledge, or 'knowns': known knowns, known unknowns, and unknown unknowns.[4] Gervasi et al. [7] suggested that there is a fourth type of knowledge:

> *"An unknown known is knowledge that a customer holds but which they withhold from the analyst."*

We note that this 'withholding' is likely to be unintentional. Tacit knowledge, then, fits that definition, i.e. tacit knowledge is an unknown known [7]. Table 1 presents the four types of knowns, considering two important roles in RE, as Gervasi et al. [7] suggested: the system analyst/developer, and the user. Some knowledge is held by both developers and users (known knowns), whereas other knowledge is known to only one but not the other, e.g. unknown knowns represent knowledge held by users, whether they are aware of it or not, but unknown to developers.

[4] These different types of 'knowns' map very well to Phillip Armour's "Orders of Ignorance" published two years prior, in 2000 [1]. This might be a rare unintended instance where SE research has had an impact on global political rhetoric.

**Table 1.** Developers and users' knowns and unknowns (based on Gervasi et al. [7] and Sutcliffe and Sawyer [20])

| | | Analysts and Developers | |
|---|---|---|---|
| | | Known to developers | Unknown to developers |
| Users | Known to users | **Known Knowns**: relevant knowledge that users know and that can be articulated for software developers | **Unknown Knowns**: relevant knowledge that users know (whether consciously or without realizing it), but which is not yet articulated and thus not known yet to software developers. |
| | Unknown to users | **Known Unknowns**: relevant information that developers are aware of (know), but which they don't know yet. Users may be unaware of this knowledge, or have forgotten it. | **Unknown Unknowns**: potentially relevant information, but both developer and user are unaware that it is missing. Developers lack relevant domain knowledge, and users are unaware of the knowledge that they rely on. |

The RE field has discussed different requirements elicitation techniques at length [4,8,20] and it lies beyond the scope of this paper to present a full discussion. Commonly discussed techniques are interviews, workshops, scenarios, and observation [4,8,20,22]. Observation is often mentioned as a part of conducting ethnographic studies; several papers have discussed ethnography or observational approaches to support requirements elicitation and design [9,11,19]. Early studies proposing to integrate ethnography for RE recognized that traditional techniques *"do not take into account actual work practices"* [19]. While there has been some fruitful discussion and analysis of observational methods to uncover 'unknowns,' a few issues seem to remain. For example, ethnographically-informed or observational methods for RE have been suggested to require considerable resources, time in particular, making them less attractive. Further, other issues associated with ethnographic research is that it may suffer from ambiguous interpretation [20], a lack of technical competence of ethnographers [3], and the serendipitous nature of identifying new requirements through ethnography [20], i.e. the reliance on luck. Finally, the number of studies that evaluate the use of observational approaches including ethnography for requirements engineering has remained limited, despite several important contributions in the 1990s and early 2000s [11,10,9,3]. One might wonder, given the drawbacks listed above, whether organizations should bother with observational approaches. A lack of experience reports on the use of observational or ethnographic techniques hinders organizations in deciding whether this approach is worth the considerable cost. There seems to be an acceptance that there is no advantage in any specific technique over the use of structured interviews [8,20], and so an open question is: what value does observation offer in a requirements engineering context?

Thus, the goal of this experience report is to highlight the importance of observation to identify unknown knowns and unknowns, and report lessons learned from the field, in a domain that hitherto has not been studied in this context. Several of the seminal

papers in the software engineering and requirements engineering literature reported on an air traffic control system, which represents a very specific setting whereby its users operate in a fixed location. This experience report focuses on surveyors who inspect ships for certification, necessary to allow them to operate in international waters. Surveyors, unlike air traffic controllers, operate in a different setting *every single day*. This makes characterizing these actors' work environment more challenging as each ship is unique and thus it is important to recognize the varying work settings of these experts. We illustrate how developers, who had used traditional requirements elicitation techniques such as interviews, were struggling to understand these expert users, and indeed had not gained important insights that we classified as unknown knowns and unknown unknowns (see Table 1). We conclude by juxtaposing our findings with prior literature, adding clarifications and commentary, and identify some implications for practice.

## 2 Methods

This experience report draws on data collected from different sources at DNV, a major service provider in the maritime sector. As researchers of the SINTEF Digital Process Innovation group, we are involved in an ongoing project with DNV focused on Digital Transformation, which provided the backdrop of this investigation. In the remainder of this section we describe DNV and procedures for data collection and analysis.

### 2.1 Description of DNV

DNV is a leading service provider in the maritime sector, with about 3,700 employees operating globally. DNV's core business is compliance verification of vessels (ships of any size); successful verification leads to issuing of necessary certificates that vessels require in order to secure marine insurance and sail and operate in international waters. Certificates are normally issued annually, with a more thorough five-year survey. Vessels are costly to run; therefore, they are continuously in operation. Surveys are typically conducted during visits to ports or shipyards, when vessels load or unload cargo, or undergo maintenance. Every survey job is tailored to the unique characteristics of a vessel, and its operation plan in order to reduce the interruption to normal operations. This means that survey procedures are often broken down into parts, with each part of the survey potentially being conducted in a different port.

DNV is currently modernizing its survey support system, which surveyors use to conduct and manage surveys. The system is used for planning survey jobs, document compliance, reporting of 'findings,' (that is, issues that require fixing before compliance can be signed off), looking up a vessel's history, and issuing of certificates. The current desktop version for Microsoft Windows was released in 2004, and at the time of data collection, DNV was developing a new web-based solution to allow continuous development of new features. Development is organized as an in-house project with a release date when the new solution goes live and the old system shuts down. DNV employs approximately 1,000 surveyors globally who are the primary users. This group of users tend to dislike new digital tools – or in the words of one surveyor, *"we don't like change."* DNV management was concerned that if the new system gained a bad

reputation, the cost would rise dramatically, possibly outweighing the benefits of the new system, requiring significant resources to overcome resistance in adoption. In other words, management put a premium on developing a system that pleases its intended users.

### 2.2 Data Collection and Analysis Procedures

We collected data during a nine-month period; data collection and analysis were interleaved, and followed procedures described by Seaman [18]. The data collection activities included semi-structured interviews and several site visits for observation of surveyors at work. The first site visit for observation was treated as a pilot study, and from this we gained valuable insights into how this group of users interacts with software technology and a general understanding of their role; based on this we designed observation guides and semi-structured interview guides.

Interviews can be a valuable source of information as it allows in-depth conversation with experts, but it is only one of many potential methods to collect data in field studies [13]. Interviews fall in a category of methods that Lethbridge et al. have labeled *inquisitive* techniques [13], in that a researcher must actively engage with interviewees to get information from them. A second category is *observational* techniques, which includes observation of professionals. Both types of techniques have benefits and drawbacks; interview data may be less accurate than observational data, but observational techniques may introduce the Hawthorne effect, whereby professionals' processes change when they are observed [13].

A total of nine interviews were conducted: four software developers, one manager, one implementation manager, and three surveyors. The focus of these interviews was to develop an understanding of the purpose of the new system and how developers elicited user requirements. All interviews were transcribed, resulting in 105 pages of text. Following the interviews, we conducted a total of seven observations. We observed three more surveyors for two days each, and three surveyors for one day each. Three of the surveyors were situated in Norway and four in the Netherlands.

The first two authors conducted observations of seven surveyors. These onsite activities were organized in collaboration with DNV's central scheduler who assigns jobs to surveyors. The site visits involved shadowing the surveyors for the full day; this included accompanying surveyors during inspection of vessels, including crawling through narrow storage tanks, climbing crane towers, as well as driving for hours to reach remote ports during which surveyors could also have phone calls with colleagues, and having lunch together. Our impression is that the surveyors appreciated the opportunity to show their work practices and expressed themselves freely. The researchers conducting observations were dressed similarly to crew and surveyors, including all the required Personal Protective Equipment (PPE) (including safety helmet, ear muffs, safety shoes, etc.). In a way, we were more like apprentices than researchers. Research notes and pictures were constantly captured over the course of data collection, and reflections were written immediately afterwards, resulting in 59 pages of notes produced and about 100 photographs of surveyors in action (see Figure 1).

The first and second authors jointly analyzed the data and immersed themselves in the material. A word processor was used for both open-ended coding and memoing [18]. Examples of labels include:

- "use of phone calls, not chat, to maximize bandwidth of communication and realtime feedback"
- "surroundings force surveyors to take breaks during their work day"

These two labels were grouped in a theme "adapting to surroundings." After we completed the data analysis, we used member checking, a procedure to assess the validity of our findings by presenting them in a workshop involving surveyors and DNV management, and adjust any misapprehensions. Overall, their response was confirmative.

## 3 Findings

This section presents the key findings. We first discuss requirements elicitation practices at DNV; the remainder of the section is organized using the framework presented in Table 1; a summary of findings is presented in Table 2.

It should be clear that the 'users' in this context are domain experts, namely, surveyors of the DNV organization, who have very extensive experience; the term 'users' does not therefore apply to other types of users.

We do not discuss Known Knowns in further detail, because that is knowledge shared among developers and surveyors alike. One example of this is that surveyors preferred to keep the new system as similar to the old one, and developers were aware of this and tried to accommodate this.

### 3.1 Requirements Elicitation at DNV

The team we interacted with did not have any dedicated requirement engineers; requirements therefore were elicited by the developers. Development teams relied mainly on two methods to capture the surveyors' requirements.

1. **Workshop.** A three-day workshop with a user representatives group of 50 surveyors face to face in the early phases to gather as much information as possible about how surveyors work and interact with technology. One developer explained their focus on the gap between current features and what surveyors need and expect: *"We were discussing current solutions and what they [the surveyors] miss."* Following the workshop, a UX-designer created user stories based on the results.
2. **User tests.** Variants of one-on-one test sessions virtually on Microsoft Teams to test usability and functionality developed from the user stories gathered in the workshop. One surveyor in the user representative group explained:
   > *"I share my screen and they sit and take notes along the way or ask [questions]. We did tests where I kind of got instructions on what to do (...) and tests to check if I intuitively could find what to do."*

   These tests were facilitated by the UX-designer and ranged from strictly orchestrated, to tests where the user was given a task and encouraged to explore the system on their own to solve it. Tests were conducted every 2-3 months with the same selection of surveyors as the one attending the first workshop.



**Fig. 1.** From left to right: planning the survey, photographing issues, surveying in difficult environments, paper artifacts remain important, and one of the researchers on-site. No photographs could be taken on tankers as we had no explosive-proof camera.

### 3.2 Known Unknowns: What Developers Know They Don't Know

Developers were aware that they did not know certain aspects. To better develop and gain insights into these known gaps in their knowledge ('Known unknowns') developers had continued access to the user representative group during development. This proved useful when developers sought to ask for clarifications while reading user stories and developing features. One of the developers explained that: *"Some assumptions that we've had before proved slightly different."*

**Table 2.** Summary of Findings

| | Known to developers | Unknown to developers |
|---|---|---|
| Known to users | **Known Knowns**:<br>– Surveyors preferred to keep the new software as similar as possible to the old one, which developers knew and tried to accommodate. (Not discussed in the findings section, because this is unproblematic knowledge.) | **Unknown Knowns**:<br>– Surveyors rely on "gut feeling" and "on the go" decision making.<br>– Surveyors prefer to discuss issues over the phone with colleagues, instead of in writing (e.g. chat or email), ignoring the 'chat' function that was designed for this.<br>– Surveyors spend much time interacting with people onboard of vessels, which is essential for a successful survey, but is hard if not impossible to capture. |
| Unknown to users | **Known Unknowns**:<br>– Developers frustrated with an inability to test their assumptions.<br>– Developers lacked domain knowledge and realized they were missing information when merely talking to surveyors.<br>– Developers envisioned the user according to how they understood them from the workshop and interviews, filling in blanks using their own logic rather than by asking the surveyor, and their experience might be limited. | **Unknown Unknowns**:<br>– Divergence between observational and interview data: surveyors simplified, generalized, and abstracted when talking about their job, but reality is different.<br>– Surveyors must constantly adapt their workflow to changing circumstances; the survey process is not straightforward and is tailored to the context.<br>– The application to report findings (issues that require fixing) technically works, but is not used as originally conceived by designers due to inconvenient menu navigation and illogical object naming. |

Developers described this combination of observation during user tests on the one hand, and the ability to contact surveyors directly for follow-up questions on the other hand, as 'ground-breaking' because they had not had such close contact with users before. At the same time, they sought to acquire an even deeper insight into the user's context and observe surveyors use their software in the real world, to understand *"How does this [software] relate to how they actually work?"* Developers were aware they lacked this understanding, and expressed a preference to visit the world of surveyors and conduct observations themselves. One developer shared that: *"I have not been on a tanker before, so I have no idea what things really look like."*

Developers argued they could not obtain all the essential information about user needs due to a lack of basic domain knowledge of surveying. One of the developers recognized that context information was crucial to gaining a deeper understanding of the survey process in practice:

> *"I would like to get on a boat and see what the actual work process looks like [...] there is a lot they [surveyors] cannot include when they explain it in the office, versus when you are actually out physically with them."*

At the same time, observing a surveyor in real life is not without challenges, not least of which include the cost associated with site visits as well as the purchasing of prerequisite PPE, and the cost associated with lost developer productivity for the time they travel. One developer highlighted:

> *"We have always requested that we visit a ship so that we can actually connect the dots between our domain knowledge versus what we actually see in practice."*

Developers were frustrated that some of their work had to rely on assumptions that were impossible to test through traditional methods such as workshops and interviews. It seemed there was a shared recognition among the developers that if they could not test these assumptions, the software would not fulfill its potential.

### 3.3 Unknown Knowns: What Developers Don't Know, But Users Do

There was also a category of knowledge that developers were not aware of, but the surveyors were. All surveyors we shadowed were highly experienced in their role. When boarding a vessel, they would quickly gain a 'feeling' of the vessel's condition, as they had learned to recognize subtle clues of technical problems ("findings") through observations and conversations after years of experience. Subtle clues include the freshness of the paint, the general tidiness of the deck, the condition of the lights, and signs of stress among the vessel's captain and crew. By piecing together these clues, surveyors adapted the survey job to uncover the most crucial findings. For example, one of the surveyors we followed decided to check all so-called ex-lights (lights certified for explosive environments) after having been only minutes on board a vessel. The surveyor then continued to make numerous other findings. When we asked why he decided to go straight for the lights, he said that it was a combination of experience and pieces of information he gathered when boarding the vessel, concluding: *"I get this gut feeling."* He did not need to spend time analyzing what to focus on in his survey but intuitively made this decision *"on the go"*—the checklist or survey support system did not prompt the surveyor. This was knowledge he found impossible to describe to developers, arguing: *"You can only learn it through years of experience."* We made similar observations with other surveyors, all of whom shared the same explanation.

Although surveyors tried to explain this type of knowledge during interviews, it was not until we observed this ourselves, while attending surveys on vessels, that we gained an adequate understanding of how such knowledge impacted the way the survey was conducted. It had remained unknown to us as researchers and to the developers.

Because the surveyors struggled to communicate many critical aspects of their work to others (non-surveyors), they felt they were unable to articulate their needs clearly to

developers. Previously, DNV had digitalized the old paper-based checklists to relieve surveyors of hours at the printer. Despite knowing most checklists by heart and rarely printing them, surveyors welcomed this improved accessibility. However, whereas the old checklists automatically marked newly added check items in red, this feature disappeared with the introduction of a new system; this forced surveyors to search for this information that was now 'hidden' within hierarchies of application menus. Surveyors agreed that a lack of understanding of surveyor work among software developers was the reason for this shortcoming.

Another crucial part of surveying, that remained completely unknown to developers, is how surveyors establish relations with the captain and crew upon entering a vessel. This is important because both captain and crew act as 'gatekeepers' as well as facilitators for the survey. Good working relations results in a smoother survey because the captain and crew willingly support the surveyor in accessing the vessel's different parts, i.e. by opening hatches, stopping maintenance work, clearing gas tanks, etc. They are powerful allies because they can provide flexibility to the surveyor. Surveyors and crew constantly negotiate which survey activities are convenient based on the current situation on the vessel. For instance, when a surveyor planned to inspect tanks, a cooperative relationship with the Chief Officer gained him increased access and guidance during the survey.

Establishing good relations happens in social situations. Typically, when surveyors board a vessel they go straight to the bridge and meet the captain and First Officer. Coffee is offered, sometimes cakes, and usually a polite conversation ensues about for example the vessel's history and mutual acquaintances. They then move to planning the day and negotiating what surveyor activities are possible considering the vessel's operations that day. Drawings of the vessel and survey-checklists are commonly central artifacts for achieving a shared vision amongst them, because they are able to show and tell, pointing to details as a basis for discussions (see Fig. 1). These artefacts are found either on the surveyor's computer screen or printed by the captain to make it easier for them to gather around.

Social situations, like these, remain unknown to developers because surveyors perceive this to be an informal part of the survey process, and as irrelevant to developers:

> *"I spend half my time going around the vessel talking to people. That human part is not very well captured [in work instructions, procedures, etc.]."*

This becomes a challenge because developers do not know that the software they develop is a critical artefact in establishing good relations between surveyors and captains.

Another example of unknown social interactions occurs when surveyors talk on the phone. Surveyors are dealing with problem solving in complex surroundings and often need to discuss their situation with colleague surveyors. In a time-pressed world they reach out for colleagues because they are able to explain their problem and surroundings within minutes, let their colleagues ask questions, and interpret the situation together to agree on how to proceed. Such conversations contain valuable and almost-instant information about problem-solving that DNV attempted to capture by introducing a chat function. By capturing these conversations through chats in the application, knowledge could be saved and processed to benefit other surveyors experiencing similar problems. However, surveyors only used the chat for straightforward and simple issues, leaving

the more complicated issues for phone conversations, meaning the most valuable and interesting conversations were never captured. One surveyor argued that:

> *"It [a phone call] creates fewer misunderstandings, less dissatisfaction, more understanding and makes it easier to reach an agreement. You don't waste time writing e-mails"*

We observed such conversations between surveyors and actors like captains, engineers, managers, superintendents, ship owners, authorities, and colleague surveyors. Characteristics of such phone calls remained unknown and indeed invisible to developers.

These were some examples of critical pieces of expert knowledge about surveyors' daily jobs, constraints and requirements that developers simply were not aware of, and that surveyors struggled to articulate during requirements elicitation.

### 3.4 Unknown Unknowns: What Neither Developers Nor Users Know

Finally, there was a category of knowledge that neither developers nor the surveyors were aware of, the unknown unknowns. The surveyors' workflow was highly dependent on, and tailored to a complex context with constantly changing circumstances onboard the vessels. This was an important issue for developers to understand, yet they were not aware of this, and nor were surveyors readily aware of this as they described their work in interviews. For example, surveyors were forced to take short spontaneous breaks caused by changing circumstances on deck. On one occasion, they had to wait for a crew performing gas measurements before the surveyor could enter a tank—strict protocols are in place before people may enter certain hazardous areas of a vessel. The loading and unloading of cargo also affected the survey activities that could be performed; bunkering (refuelling of a vessel) could also limit certain activities, such as a black-out test that tests back-up generators. Surveyors would use these breaks to capture findings. Although surveyors were expected to use the application to do this, we observed that this involved large numbers of clicks and taps when navigating menus to register findings. Surveyors found this too cumbersome, because breaks were usually short and unpredictable. In addition, several surveyors found it challenging to locate the correct 'object' in the application because their naming did not always make sense to them. Therefore, they avoided using the application during inspections and instead preferred to register their findings manually, for example by taking photographs of issues, or handwritten notes.

Although the application seemed to be an excellent digitalization effort in theory, it failed to support the experts in practice because it was incompatible with how they performed their job in a complex and unpredictable environment. The surveyors perceived that the system failed because their context was too complex to describe all of its aspects to software developers. Interestingly, when we compared our observational data and interview data, we discovered significant divergences in how surveyors *said* they worked and how they *actually* worked. During interviews, surveyors would generalize and simplify how they worked, leaving out details and abstracting away aspects that are hard to comprehend without field experience. During observation, however, they referred to situations they participated in as a starting point for more detailed explanations. They were not able to translate their expert needs to software requirements. They acknowledged that these requirements remained unknown to them and that developers would

**Table 3.** Contributions and Implications

| Insights from prior literature | Findings of this Study | Implications for Practice |
|---|---|---|
| Observation and ethnography are resource-intensive activities. Early scholars proposed 'quick and dirty' approaches" [9]. | The gap between developers and users may be too large to bridge; observing surveyors even for a few days provided valuable insights that are difficult to articulate. | The cost associated with letting developers go into the field for even a week may well be worth it, potentially improving the quality of requirements analysis. |
| Observation and ethnography depend on 'luck' and serendipity to identify requirements [20]. | Observing expert surveyors at work for even a few days provided deep insights into the constraints and work practices. | Observation is not only about identifying requirements and features, rather, it can be valuable for developers to understand the real-world context and constraints (see Fig. 1). |
| Integrating ethnographic observations into structured methods of requirements analysis is very challenging [19]. The reductionist character of RE (focusing on components, data flows, processes) is not compatible with ethnographic inquiry [3]. | Surveyors prefer ad hoc communication *outside* the system's functionality (rather than built-in features e.g. chat). The social interactions and goodwill between surveyor and captain/crew are essential for a successful survey, which cannot be captured in system features. | Rather than seeking ways to integrate observational findings into requirements analysis, consider system boundaries and develop a good understanding of the social context where systems are implemented. |
| Experts find it difficult to articulate their expertise during requirements analysis [19]. Ethnography recognizes work activities as they are actually conducted, rather than some idealized version of it [10]. | Surveyors frequently rely on 'gut' instincts developed over years of experience. Surveyors are often assigned jobs on short notice, and have to adjust surveys based on continuously changing circumstances. | Not all work practices can be digitalized. The valuable experience and gut instincts cannot be replaced with a rigid workflow. Develop systems that empower experts, rather than aiming to digitalize existing workflows. |

benefit by making observations in the real world. One of the surveyors proposed that *"They [developers] have to come out here and see for themselves."*

In sum, both developers and surveyors recognized the need for developers to observe the world of surveyors to better understand the needs of this group of expert users.

## 4 Discussion and Conclusion

Most prior work in the software engineering domain was conducted by a relatively limited number of authors, detailed many insights from a select number of case studies, in particular studies of air traffic controllers [10]. Several other studies are situated in

the HCI and CSCW communities in the 1990s and early 2000s, recording lessons for designing interactive systems that became increasingly popular in the 1990s [2].

In this paper we report a number of insights (see Table 3). For example, whereas prior literature has suggested the high cost of ethnographic or observational studies might preclude organizations from using these strategies, we would argue that in some domains the gap between developers and users is simply too large to bridge. Even a few days of observing surveyors at work provided extensive insights that could have saved many hours of developer time. It is important at this point to distinguish between 'observation' and 'ethnography'; the former being a technique that we focused on primarily, whereas the latter is a more encompassing strategy that would take far more time.

A second issue with observation is that it depends on serendipity and 'lucky' circumstances that would lead to identifying new requirements. While we agree this is an issue in *identifying* requirements, it is less of an issue to *understanding* the nature, context, and constraints of an expert's daily job. We argue this is an important distinction to keep in mind when planning site visits for developers or requirement engineers; the goal of observation then becomes one of "walking in the user's shoes," to understand the system-in-action context before attempting to capture requirements.

Analysts and developers are trained professionals who look at systems through a lens of the *affordances* that technologies offer, but this too has a reductionist 'smell': looking at technologies and how they "map" to possibilities, rather than user preferences. It is possible, of course, to add a chat function that seeks to capture potentially valuable conversations—but this ignores users' *preference* to *not* use system features, but use simple means such as a phone for a direct and possibly private conversation.

Finally, experts are known to experience difficulty articulating their knowledge, certainly when this includes dependency on their gut instincts that developed over years. No guidelines, handbook, or IT system can replace this. This leads us to argue that, perhaps, not all work practices can (and should) be digitalized. In a way, we are touching upon the boundary of a current popular trend in the IT industry, namely a search to digitalize everything. While we do not deny that software systems can greatly improve our lives and productivity, care should be taken to understand how software solutions can *support* our work practices rather than replace them.

In conclusion, software systems are designed for expert users, but in some domains these experts have very extensive tacit knowledge. Drawing on previous insights on tacit knowledge, we reported experiences with some of the issues in a domain where changing circumstances, users' 'gut instincts' and extensive experience, and physically challenging environments are important issues when seeking to identify unknown knowns and unknowns.

**Acknowledgements.** We thank the participants of this research for sharing their insights, and the surveyors whom we observed for their patience and insights. We are grateful to DNV for funding this research together with The Norwegian Research Council (grant number: 309631). For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

## References


1. Armour, P.G.: The five orders of ignorance. Communications of the ACM **43**(10), 17–20 (2000)
2. Coad, P., Yourdon, E.: Object-Oriented Analysis. Prentice-Hall, Inc. (1990)
3. Crabtree, A., Rodden, T.: Ethnography and design? In: International Workshop on Interpretive Approaches to Information Systems and Computing Research (2002)
4. Davis, A., Dieste, O., Hickey, A., Juristo, N., Moreno, A.M.: Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review. In: 14th IEEE International Requirements Engineering Conference (RE'06). pp. 179–188. IEEE (2006)
5. De Marco, T.: Concise notes on software engineering. Yourdon inc. (1979)
6. Gacitúa, R., Ma, L., Nuseibeh, B., Piwek, P., De Roeck, A.N., Rouncefield, M., Sawyer, P., Willis, A., Yang, H.: Making tacit requirements explicit. In: 2nd International Workshop on Managing Requirements Knowledge. pp. 40–44 (2009)
7. Gervasi, V., Gacitua, R., Rouncefield, M., Sawyer, P., Kof, L., Ma, L., Piwek, P., de Roeck, A., Willis, A., Yang, H., Nuseibeh, B.: Unpacking tacit knowledge for requirements engineering. In: Managing Requirements Knowledge, pp. 23–47. Springer Berlin Heidelberg (2013)
8. Hickey, A.M., Davis, A.M.: Elicitation technique selection: how do experts do it? In: Proceedings. 11th IEEE International Requirements Engineering Conference. pp. 169–178 (2003)
9. Hughes, J., O'Brien, J., Rodden, T., Rouncefield, M., Sommerville, I.: Presenting ethnography in the requirements process. In: Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95). pp. 27–34. IEEE (1995)
10. Hughes, J.A.: Ethnography, plans and software engineering. In: IEE Colloquium on CSCW and the Software Process. IET (1995)
11. Hughes, J.A., Randall, D., Shapiro, D.: Faltering from ethnography to design. In: Proceedings of the 1992 ACM conference on Computer-supported cooperative work. pp. 115–122 (1992)
12. Jensen, R.W., Tonies, C.C.: Software engineering. Prentice-Hall (1979)
13. Lethbridge, T.C., Sim, S.E., Singer, J.: Studying software engineers: Data collection techniques for software field studies. Empirical software engineering **10**, 311–341 (2005)
14. Levitt, S.: Episode 4. Ken Jennings: "Don't neglect the thing that makes you weird". People I (mostly) Admire. Podcast (3 October 2020)
15. Mead, N.R.: A history of the international requirements engineering conference (RE) RE@ 21. In: 21st IEEE International Requirements Engineering Conference. pp. 21–221 (2013)
16. Paetsch, F., Eberlein, A., Maurer, F.: Requirements engineering and agile software development. In: Proceedings Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. IEEE Comput. Soc (2003)
17. Polanyi, M.: The tacit dimension. In: Knowledge in organizations, pp. 135–146. Routledge (2009)
18. Seaman, C.B.: Qualitative methods in empirical studies of software engineering. IEEE Transactions on software engineering **25**(4), 557–572 (1999)
19. Sommerville, I., Rodden, T., Sawyer, P., Bentley, R., Twidale, M.: Integrating ethnography into the requirements engineering process. In: [1993] Proceedings of the IEEE International Symposium on Requirements Engineering. pp. 165–173. IEEE (1993)
20. Sutcliffe, A., Sawyer, P.: Requirements elicitation: Towards the unknown unknowns. In: 21st IEEE International Requirements Engineering Conference (RE). pp. 92–104 (2013)
21. Tracy, K.W.: Software: A Technical History. ACM (2021)
22. Zachos, K., Maiden, N., Tosar, A.: Rich-media scenarios for discovering requirements. IEEE software **22**(5), 89–97 (2005)