Towards Well-Founded and Richer Context-Awareness Conceptual Models

Boris Shishkov[1,2,3,4(✉)] and Marten van Sinderen[5]

[1] Faculty of Information Sciences, University of Library Studies and Information Technologies, Sofia, Bulgaria
[2] Faculty of Technology, Policy and Management, Delft University of Technology, Delft, The Netherlands
[3] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria
[4] Institute IICREST, Sofia, Bulgaria
b.b.shishkov@iicrest.org
[5] Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands
m.j.vansinderen@utwente.nl

Abstract. We observe that context-aware systems currently developed in one domain or another are mostly technology-driven, and not so much user-centric. They are often not based on a thorough analysis of the effects they produce when interacting with their context, especially regarding the contribution of these effects to user needs. We argue that a conceptual framework is needed to support such analyses. In this paper we identify the concepts necessary to define important structural aspects of a context-aware system and its context, and to formulate generalizations about effects of the interaction of the context-aware system and its context related to user needs. Using this conceptual framework, we classify context-aware systems in terms of the kinds of context assumptions that we can make at design time, and we discuss several threats to validity of a context-aware system. We believe that the proposed conceptual framework can help to better assess the utility concerning a context-aware system design. We use various examples of context-aware applications to illustrate our ideas.

Keywords: Adaptive service delivery · Context-awareness · Conceptual modeling · Architectural structure · User needs · Utility analysis

1 Introduction

Context-awareness is receiving much attention in numerous application domains - from mobile health monitoring [12] to drone-driven monitoring in areas affected by disruptive events [13]. We argue that even though those applications are useful and well-reflected in corresponding R&D materials, scientific papers, and project documentation, they are often technology-driven and not driven by user needs. We argue that there is a lack of solid conceptual foundations that are rich enough to support top-down design of context-aware applications. In the current paper, we propose a conceptual framework that serves this purpose.

B. Shishkov (Ed.): BMSD 2021, LNBIP 422, pp. 118–132, 2021.
https://doi.org/10.1007/978-3-030-79976-2_7

Context-awareness essentially concerns adaptive service delivery [9, 16], for which three adaptation perspectives are possible, viz. serving (i) user needs; (ii) system needs; and (iii) public values. Although these perspectives are all equally important, for our conceptual framework, we currently only consider (i).

We claim that our conceptual framework helps to support user-centric design by making explicit which threats to utility exist and providing the concepts to discuss and resolve these threats at design time. Here, we consider a system to have utility (usefulness) if it provides services that satisfy the user needs. Although we cannot measure utility at design time, we can justify design choices with "satisfaction arguments": reason that those are the best among alternatives, using logical arguments that consider the user needs. As part of the framework, we also propose a classification of context situations in terms of how well context can be foreseen and defined at design time. In cases where context situations cannot be completely or properly defined, machine learning approaches can be used to detect (or predict) context situations related to user needs. This opens the possibility to extend the framework further to assess the suitability of machine learning methods [14, 15] with respect to their usefulness as it concerns context-aware systems. We plan this as future work.

The remaining of the current paper is structured as follows: Sect. 2 presents a historical perspective on technological developments that led to the context-aware systems of today and discusses the technological bias of many context-aware systems. In Sect. 3 we present our proposed conceptual framework. In Sect. 4, we partially exemplify our proposal. And in the end, in Sect. 5, we discuss the framework and its limitations as well as our plans for future work.

2 Background

In this section, we firstly mention some technological developments that led to context-aware systems and secondly we consider the technological bias of a number of such systems.

2.1 Historical Perspective

Back in the 1980s and 1990s, computers and information systems were quickly gaining popularity [16]; the behavior of such systems was initially fully user-input-driven, and any change of use/needs had to be explicitly indicated by the user [1]. In dynamic environments with corresponding changing user needs, this is considered a drawback. Automated adaptation of system behavior to context changes as well as seamless service provisioning only became possible in the new millennium, when three useful developments took place, namely: (i) Miniaturization of computers leading to mobile computing devices [2]; (ii) GPRS/wi-fi connectivity of these devices, allowing to receive support from more powerful computing systems in different situations – while walking, while visiting "another place", etc. [3]; (iii) Sensor technology embedded in the devices, enabling the measurement of physical variables and derivation of the user situation [4].

This led to the emergence of context-aware computing, in the first decade of the new millennium, assuming the possibility to adapt the delivery of ICT (Information and

Communication Technology) services to the situation of the user [5]. At the same time, we have witnessed developments in the area of autonomic computing [6], featuring the self-management of computing resources. Finally, value-sensitivity [7, 8, 18] has more recently been proposed, for the sake of using adaptation of service delivery for supporting particular relevant public values, such as privacy, accountability, and transparency.

We currently observe context-aware applications that are developed in various domains. Most of those applications are technology-driven (a "bottom-up" perspective), aiming to show new technology applications, without a thorough understanding of the effects produced by the corresponding context-aware services on the user and his/her environment and their contribution to context-dependent user goals (a "top-down" perspective).

2.2 The Technological Bias

Pioneering researchers in the area of context-awareness have definitely improved our understanding of the notion of **context** and made serious progress in the development of **context-aware applications** [1, 19, 28, 29]. We argue nevertheless that often: (i) there is a *bottom-up* approach to application development; (ii) the challenge of tackling situations, when context states cannot be foreseen at design time, is not explicitly considered. The same holds for many R&D context-awareness projects, such as CyberDesk [30], AWARENESS [12, 32], and SECAS [33]. In these works: **user-centricity** does not seem to play a major role in the design; the consideration of *user needs* is not an explicit part of the design cycle.

The useful survey of Alegre et al. [22] is mainly focused on the development of *context-aware applications* as well as on the consideration of *public values*. The same holds for the works of Alférez and Pelechano [23] – they consider the dynamic evolution of *context-aware* systems, the development itself, and the relation to *web services*. The latter holds also for the *service-orientation* perspective as proposed by Abeywickrama [24]. All these works take a primarily **technology-driven perspective** and are less concerned with the **user perspective**. The same holds for other works touching upon the *adaptive delivery of services*, always considered in a *bottom-up* perspective, featuring *decision-making* [25], *safety of stakeholders* [26], and *routing* [27].

Exceptions can also be mentioned. For example, in [20], the authors propose a modeling approach (based on Causal Loop Diagrams) for understanding the context in relation to *user needs*/*goals*, independent of any technology. Furthermore, in [31], the central role of *human users* is acknowledged and information modeling for the *context-aware* system is based on *knowledge descriptions* using *ontologies* and *rules* [21]. Nonetheless, a conceptual framework for understanding the nature of *context-awareness* and analyzing potential issues with *context-aware* systems from a *user perspective* is lacking.

3 Conceptual Framework

We need a *conceptual framework* specifically to be able to assess, *at design time*, the utility of a context-aware system in an intended *context*. With such an assessment the designers and other stakeholders can decide whether the proposed system is ready for transfer to practice, or whether another design cycle for further improvement should be entered. In this way it is possible to reduce the risk that the *context-aware* system, once implemented in its *context*, does not fulfil the expectations of the stakeholders, and especially of **end-users**. As stated before, the purpose of a *context-aware* system that is transferred to practice is not to be technologically innovative but to better serve the user needs.

3.1 Context-Awareness

As a problem theory for *context-aware* systems we postulate that *end-users* (*users*, for short) of *information systems* often have different needs for services provided by such systems, where different needs correspond to different context situations. *Context-aware* (information) systems are a "treatment" for this problem if they can provide **context-specific services to users in accordance to their context-dependent needs**. "Context" here is the *context* of the *context-aware* system, where the former is a given (i.e., not designed) and the latter is the object of design. A *context-aware* system that is transferred to practice would interact with its context. Two kinds of interactions can be distinguished: one for *collecting data on the context* and another one - for *delivering a service* that matches the *context*. The fact that the *service* is delivered to a *user* means that the user is part of the context. This makes perfect sense, as the *part-of* relation is an essential prerequisite for the system we want to design, viz. to make a connection between what the *context* is and what a *user* needs.

We frame the design problem with the diagram in Fig. 1. The diagram shows that a **user**, being *part of* a **context**, has *one or more* **user need**s (or sets of *user needs*), where each distinguished *user need* results from a corresponding unique **context situation**. A *context* can be conceived as a *temporal composition of one or more context situations*, where each *context situation* has a unique set of *properties* that collectively are relevant to a specific *user need*. A useful **context-aware system** is able to detect the *context situation* at hand and then offers one or more **situation-specific services** that satisfy the *needs* of the *user* being in, or experiencing, that situation.

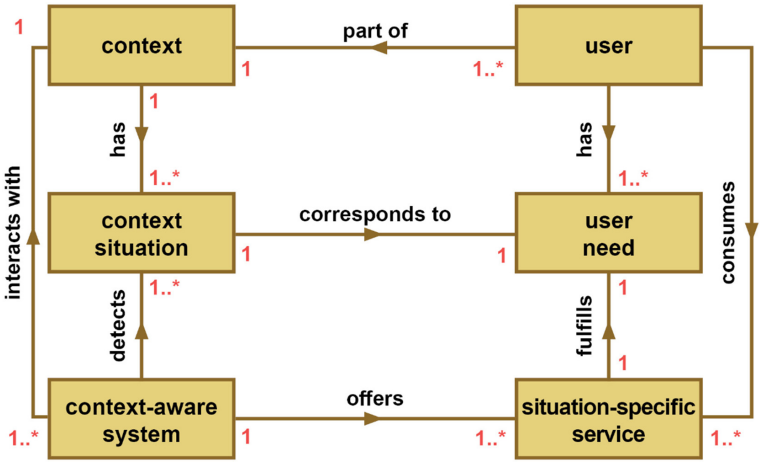

Fig. 1. Framing the problem of context-awareness

3.2 Context-Aware System

Many *architectures* of *context-aware systems* have been proposed in literature [5, 36, 37]. Figure 2 shows an architectural structure that identifies the main components and their relations. The main components are:

- **Data Acquisition & Preprocessing** (DAP): Responsible for "measuring" the context using sensors, and for cleansing and aggregating the data from sensors, for the sake of obtaining a more reliable data set suitable for analysis.
- **Situation Detection** (SD): Responsible for analyzing the data set, which consists of interpreting the data set as a context model (i.e. a sensor-data-based representation of the context) and deciding whether the represented context satisfies the properties of a context situation.
- **Adaptation to Situation** (AS): Responsible for creating or selecting the capabilities that are required to provide a service that is suitable for the context situation at hand.
- **Situation-specific service Offering & Delivery** (SOD): Responsible for offering the situation-specific service and delivering the service through interaction with the context. The service delivery can involve the use of actuators; this is to control a mechanism in the context and/or a user interface, for properly interacting with a user in the context.

What is referred above as the data set and the context model, respectively, are actually *a time series, representing the context evolution over (a period of) time*. The interactions between the mentioned components explain the behavior of a *context-aware* system: The **DAP** component collects *raw data* about the *context* and passes data useful for analysis to the **SD** component. It in turn analyzes the data and informs the **AS** component whenever a new *context situation* has been *detected*. The **AS** component then makes the *capability adaptation* necessary for a new *situation* and subsequently informs the **SOD** component

that a *new service* must be provided. This component uses the adapted capability to offer and deliver the *new service* to the *context*.

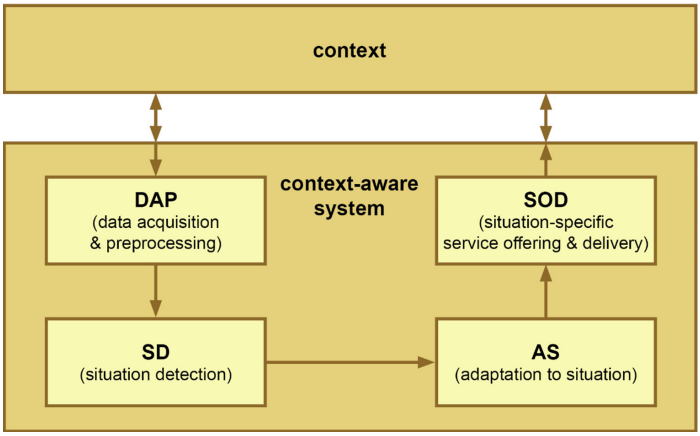

Fig. 2. Architectural structure of a context-aware system

3.3 Measuring the Context

In order to be able to better explain what the challenges are, featuring the design of **user-centric context-aware systems**, we introduce some additional concepts below (see also the diagram in Fig. 3):

We argued that a context situation has a unique set of *properties* that are collectively relevant to a specific user need. So, when "measuring" the *context*, one would actually be interested in these *properties*. For each property, one has to define one or more context indicators that can be measured. For example, the *context situation* with the *property* "hot" can be operationalized by the *indicator* "temperature". An *indicator* has one or more measurement methods. For example, *temperature* can be measured with a mechanical method (e.g. the expansion of an enclosed quantity of mercury) or with an electrical method (e.g., thermocouples). Indicator measurements, obtained with the selected measurement methods, are used to create a *context model* that focuses on the *properties* relevant to corresponding *user needs*. For *situation detection*, it is necessary to establish whether the *properties* of a *context situation* are satisfied. In the *context model* this is done by comparing *indicator* measurements with indicator norms. A *norm* is a required range of *values* of an *indicator*. For example, the *context situation* "hot" may have as *norm* for the *indicator* "temperature" the range [30 °C–45 °C]. If the *norms* of all *indicators* for all *properties* of a *context situation* are satisfied, a situation detection event for that *situation* can be generated, which then results in providing the situation-specific service.

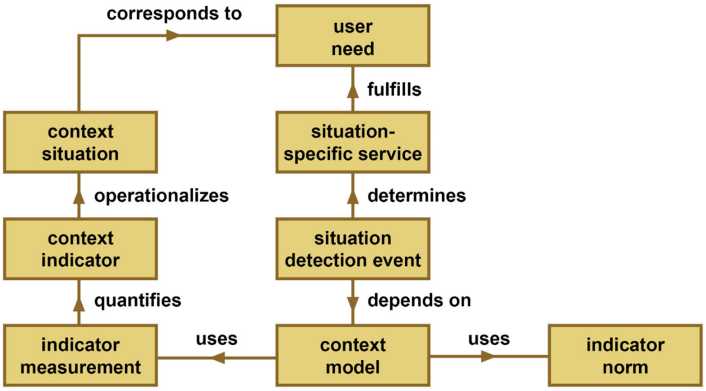

Fig. 3. Framing the design problem of context-aware systems

3.4 Context Situations

Regarding the **context situations**, the following cases can be distinguished:

- *Context situations* are defined, such that: (a-i) they can be recognized in the given *context* if they occur; (a-ii) different *context situations* cannot occur at the same time; and (a-iii) the *context* has an associated *context situation* at any time. In this case, if the *context situations* can be correctly *detected* by the *context-aware* system, there is always a situation-specific service that can be offered. As a special case, it is possible to define an "empty" *context situation* that has no corresponding *user need*, and therefore does not require any *service* offering.
- *Context situations* are defined, such that: (b-i) they can be recognized in the given *context* if they occur; (b-ii) different *context situations* cannot occur at the same time; but (b-iii) the *context* does not have an associated *context situation* at any time. This means that, even if the *context situations* can be correctly *detected* by the *context-aware* system, there may be times when the system is in an undefined state for which there is no designed behavior. To avoid this, it is possible to define a "fallback" behavior (and maybe *service* offering) that applies when no *context situation* can be *detected*.
- *Context situations* are defined, such that: (c-i) they can be recognized in the given *context* if they occur; (c-ii) the *context* has an associated *context situation* at any time; but (c-iii) different context situations can occur at the same time. This is undesirable, assuming that the *context-aware* system can only offer one *service*. Hence, either the *context situations* have to be redefined or the *context-aware* system must be able to prioritize context situations if they occur at the same time, only offering the *service* for the *situation* with the highest priority.
- *Context situations* are not (all) properly defined, such that it is not possible to recognize (some of) them in the given *context* if they occur. The reason could be that the *user needs* for different *situations* are not well understood and/or the properties to distinguish situations are not well understood or hard to define. In this case, *machine learning* could be used by the *context-aware* system to detect *context situations*, based

on *training sets* of *context models* labeled with *context situations* and/or *explicit user feedback*.

Figure 4 illustrates the first three mentioned cases (quadrant IV represents the combination of second and third mentioned ones).

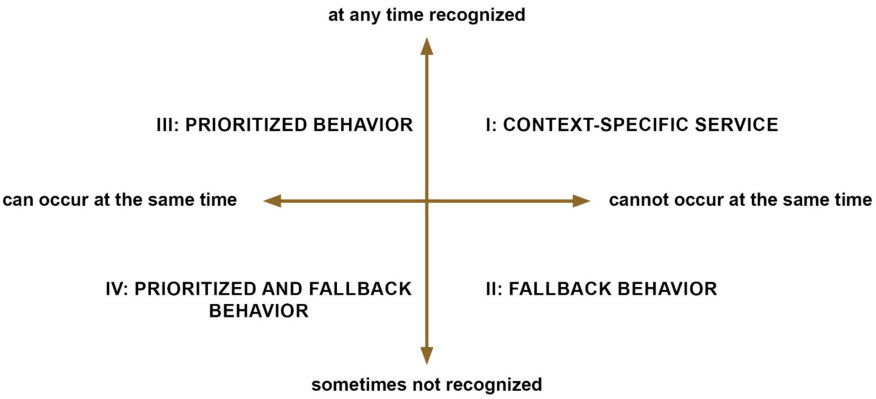

Fig. 4. Context situations and context-aware system behavior

Designers of a *context-aware* system may consider the likelihood of *context situations* and decide that certain *situations* are so unlikely that it is not worthwhile defining separate situation-specific services for them. Such situations may be covered by fallback behavior, as discussed for the second mentioned case above.

3.5 Validity Threats

If we want to claim that a context-aware system is useful, it should be possible to ascertain that the system delivers services that fulfill the user needs in context situations that occur in the context. A number of complications are possible nevertheless, threatening the validity of this claim even if the technical system design is correct.

We illustrate these threats by means of a hypothetical context-aware application, which we call the *Wellbeing at Work Coach* (WWC). More examples are provided in Sect. 4. *The WWC application monitors the interaction of an end-user with his or her computer, providing advice to the user in a message on the screen*; it is indicated to the user whenever there is a high probability that (s)he is tired, loses focus, experiences stress, and becomes less productive. The advice is, for example, to stop working and take a break, do a physical exercise, socialize with a colleague, listen to relaxing music, read a fun book, and so on. Such an advice is triggered by observing the presence of the following conditions regarding the interaction of the user with his or her computer:

- *Typing accuracy* below threshold. Typing accuracy is measured as the number of backspace key presses per key press [34].

- *Active time* above threshold. Active time, i.e. the time during which the user is considered non-stop working, is measured as the time duration of a series of mouse or keyboard events in which the time gap between two consecutive events is less than 5 min [34].
- *Stressed mouse use* above threshold. Stressed mouse use is derived based on measurements of mouse movements. Mouse movement events are used to calculate two fundamental parameters of arm-hand dynamics that capture the effect of stress [35].

Now we identify the **threats** and discuss them using the WWC application:

the situation-specific service(s) do(es) not fulfill the user need(s)
There should be a one-to-one correspondence between services and user needs, and each service should contribute to the satisfaction of a corresponding need. WWC has only one service, which is notifying the user through a message on the screen. Assuming that: (a) the user has only one need (i.e., receiving support for well-being at work); (b) the situation for this need is clearly defined; and (c) the situation detection by the application is correct, there is still a possibility that the service would not fulfil the user need. The latter is the case if *the message formulates advice that is not appealing to the user* – for example, it is possible that *the user does not want to be interrupted* if (s)he is working against a deadline. This would indicate that there is at least another user need that has to be taken into account (and that the original user need has to be scoped down to: "receiving support for well-being at work unless there is a short-term deadline").

the relationship between user needs and context situations is unclear
There should be a one-to-one correspondence between user needs and context situations, and each user need should occur only in the corresponding situation. WWC has only one user need, and one corresponding context situation that is characterized by "*a high probability that the user is tired, loses focus, experiences stress, and becomes less productive*". Whether this is an exclusive situation in which the user need occurs depends on its definition in terms of properties. WWC defines the required properties using thresholds (norms) for typing accuracy, active time, and stressed mouse use. One could define additional properties to make the 'fit' between context situation and user need better or leave out properties if they do not contribute to the 'fit' and/or are too expensive to operationalize. Furthermore, one may expect that the situation depends on the specific program(s) the user is interacting with and therefore program-specific thresholds may be necessary. In any case, thresholds should be personalized. Obviously, there is also the complement of this context situation, for which no user need exists that is of concern to WWC.

context situations are not properly defined
According to Sect. 3.4, with regard to the delivery of situation-specific services, context situations should be defined such that: situations can be recognized in the given context if they occur; different situations cannot occur at the same time; the context has an associated situation at any time. Because WWC has only one context situation (besides the abovementioned complement, which we can ignore), we only discuss *recognizing the situation in the given context*. From the WWC description, we can derive that the context situation is defined by the properties: *typing accuracy is below a threshold*, *active*

time is above a threshold, and *stressed mouse use is above a threshold*. Provided that these properties can be operationalized, the situation can be recognized in the context by comparing (processed) measurements of indicators with the thresholds (norms). Nevertheless, in more complex cases, it may be much more difficult to properly define the context situation in such a way.

the indicator(s) used for a property of a context situation do(es) not cover(s) all aspects of the property

WWC uses various indicators. *Typing accuracy* has as indicators the number of backspace key presses and the number of key presses. *Active time* has as indicator the time duration since the last keyboard/mouse event that was more than 5 min separated from its predecessor. And *stressed mouse use* has as indicator the time-stamped mouse movement events. Especially with respect to typing accuracy, one can wonder whether these indicators can be used for reliably and completely establishing typing accuracy. For example, if the user is correcting a document using a text editing program, the ratio of the number of backspace key presses and the total number of key presses would not correlate to typing accuracy. Also, the indicator for active time can be problematic, in the sense that the 5 min criterion for re-setting active time may prevent capturing actual active time (i.e., non-stop working) as experienced by the user. This criterion, representing an upper limit for the time the user has before (s)he interacts with the computer, in fact depends on the specific user task and the computer program being used. On the other hand, it is not trivial to come up with alternative indicators that are better in all circumstances.

the measurement method(s) used for an indicator do(es) not provide a reliable or proper value of the indicator

The WWC description mentions the indicators to be measured but does not cover the corresponding measurement methods. *Typing accuracy* and *active time* indicators could be measured by logging input behavior using the features available via the computer's OS API (e.g. Win32 API). The logging application may have limitations in terms of how often keyboard/mouse events are recorded, which may affect the accuracy of the indicator values. For measuring mouse movement events, a mouse motion recorder could be used that records raw-input events from the mouse. The mouse may have a limited spatial resolution, which may affect the accuracy of the measurements and ultimately the accuracy of the parameters of arm-hand dynamics.

4 Exemplification

In this section we illustrate our conceptual framework using simplified descriptions of context-aware applications.

4.1 Tele-Monitoring

A person needs to be health-monitored, such that help is provided if needed. The person is monitored from a distance, by capturing vital-sign-data through sensors.

Here, the DAP (see Sect. 3.2) uses sensors attached to the person's body, capturing vital signs, such as heart rate and blood pressure. On that basis, it is essential establishing whether the person is in a "normal" situation or in a situation assuming need for help. If, for instance, the monitored condition is epilepsy, then a combination of vital sign values would indicate a "high probability" of seizure occurrence. Hence, these are the two situations detected by the SD. On that basis, adaptations are done by the AS accordingly. For example, in the event of a "need help" situation, sensor readings would be sent in real time to a hospital, communication would be established with family or friends, and so on. These interactions constitute the delivery of the situation-specific services for the benefit of the monitored person.

Possible threats to utility here are as follows:

- Not always a "need help" situation would be captured and interpreted as precisely as to make a proper match to the actual user needs. For example, it is hard to determine whether the person would need immediate help from family/friends or is it better to wait for an ambulance that arrives with some delay.
- In case of no or poor network connectivity, fallback behavior is to be triggered but chances are small that such behavior would adequately match the user needs. For example: (i) If such a behavior would be about more and more attempts to get connected again, then a "need help" situation may be missed; (ii) If this behavior would be about just sending an ambulance, it may be that often ambulances are sent with no need for them.

4.2 Smart Lighting

A person usually needs proper illumination reflecting his or her individual preferences, in his or her living environment. Such persons are facilitated by a smart living environment, adjusting lighting accordingly in a room. The system can have a maximum number of registered users, each one identified by his or her weight.

The DAP uses a weight sensor to identify a person as a registered user when (s)he enters the room. When a person cannot be identified as a registered user (his or her weight is not close to any of the registered users), (s)he will be ignored by the system.

When a person is identified and in the room for the first time, the room lighting is turned on (if the outside illumination is below a threshold value), sticking to standard values. If the person would make any lighting adjustments, the room "memorizes" the corresponding values, associating them with the person and the time period. When the same person enters the room next time, the memorized values will be used for the lighting and any subsequent adjustments will replace (or be added to) the memorized values. Accordingly, each registered user has his or her own situation. The SD detects which situation applies, i.e. who is in the room, the AS makes adaptations according to standard or memorized values for the person, and the SOD delivers a situation-specific service for the benefit of the particular person.

Possible threats to utility here are as follows:

- If an identified person is alone in the room, being serviced by the system and another identified person enters the room, then the system is in an undefined state for which

there is no designed behavior (whether to keep servicing the person who is already in or to stop servicing him/her and start servicing the person who has just entered).
- If some of the registered persons have weights that are close to each other, it might appear that context situations are not properly defined – imagine that these persons gain or lose weight; then they may be mixed up by the system.

4.3 Mission in the Sky

A drone is flying in the sky, fulfilling a mission for the benefit of border police officers who are navigating it from the ground.

Here, the DAP uses numerous sensors. Some sensors may establish that the weather is changing (for instance), other sensors may establish the flying altitude, still other sensors may "sense" objects in close proximity (if any), the fuel/battery reserves, and so on. The drone has many alternative behaviors superposed on the predefined mission behavior. They concern various situations that might occur. Each of them is characterized by a combination of conditions that can be captured by the sensors. For example: (i) One situation might be about change in the weather; (ii) Another situation might be about getting close to an object; (iii) Yet another situation might be about reaching the "point of no return" (after which the drone would not have sufficient fuel/battery resources to come back to the ground station). The SD identifies the situation based on the sensor readings from the DAP, and the AS does the adaptions necessary for this situation. For example: (a) Algorithms running in the drone's avionic engine, adjust altitude, speed, and so on, in response to changing weather conditions; (b) Cameras continue video-recording but with applying a blurring effect, when approaching human beings, such that their privacy is protected; (c) If the drone has reached the "point of no return", the person(s) navigating the drone may be asked either to "push" the drone to immediately fly back or to update its mission (meaning that the drone would not return to the "start point" but to another location). On that basis, situation-specific services are delivered: informing the border police officers that there are no persons (potential trespassers) along the border or transmitting videos (with faces of persons blurred) featuring (a group of) persons, indicating their location, or detecting damaged border facilities, and so on.

Possible threats to utility here are as follows:

- Obviously, in such a complex mission in the sky, different context situations can occur at the same time, for example: weather may deteriorate and at the same time trespassers may be detected. Hence, prioritization is needed – whether to keep on transmitting information featuring the trespassers but assume the risk of a drone crash or adjust the flying trajectory (to save the drone) but assume the risk of "losing focus" on the trespassers.
- Also, it is possible that a context situation is not properly defined, for example: an object hitting the drone may be caused by strong wind but also be an enemy bullet. Those are sharply different situations requiring different actions but the impossibility to define the context situation complicates things.

5 Conclusions

Current context-aware systems are mostly technology-driven. For this reason, they are often insufficiently capable of delivering services that correspond to the user needs at hand (specific to a context situation). Addressing this, we have proposed a conceptual framework that is claimed to be helpful in supporting the user-centric design of context-aware systems. Further, we have made explicit which threats to validity exist, providing the concepts to discuss and resolve them at design time. We have analyzed the so called "technological bias", together with related work and the developments over time featuring context-aware systems. This was a source of inspiration for us in our proposing a broader conceptual view and an architectural structure concerning context-aware systems – both reflected in the abovementioned conceptual framework.

The limitations of our work are two-fold: (i) We have not conducted a systematic literature review; (ii) We have only used simplified examples of context-aware systems to illustrate our conceptual framework.

Future work will focus on situations that are not foreseen and can also not be accounted for in the design. Here we need risk assessments [10] and change impact analysis [11]. We expect our previous work featuring Bayesian Modeling [15] to be useful in this regard. Further, we are interested in aligning our conceptual framework to systemics [17] and public values [8]. Finally, we would carry out real-life case studies and/or interviews with experts, for a stronger justification of our proposal.

References

1. Dey, A., Abowd, G., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact. **16**(2), 97–166 (2001)
2. Krejcar, O.: Benefits of building information system with wireless connected mobile device - PDPT framework. In: Proceedings of International Conference on Portable Information Devices, Orlando, FL, USA. IEEE (2007)
3. Calvagna, A., Morabito, G., Pappalardo, A.: WiFi mobility framework supporting GPRS roaming: design and implementation. In Proceedings of International Conference on Communications, Anchorage, AK, USA. IEEE (2003)
4. Kopják, J., Sebestyén, G.: Comparison of data collecting methods in wireless mesh sensor networks. In: IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Kosice and Her-lany, Slovakia (2018)
5. Shishkov, B., van Sinderen, M.: From user context states to context-aware applications. In: Filipe, J., Cordeiro, J., Cardoso, J. (eds.) ICEIS 2007. LNBIP, vol. 12, pp. 225–239. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88710-2_18
6. Zhao, Z., Gao, C., Duan, F.: A survey on autonomic computing research. In: Proceedings of Asia-Pacific Conference on Computational Intelligence and Industrial Applications (PACIIA), Wuhan, China. IEEE (2009)
7. Friedman, B., Hendry, D.G., Borning, A.: A survey of salue sensitive design methods. In: A Survey of Value Sensitive Design Methods, 1, Now Foundations and Trends (2009)
8. Van den Hoven, J.: Value sensitive design and responsible innovation. In: Owen, R., Bessant, J., Heintz, M. (eds.) Responsible Innovation: Managing the Responsible Emergence of Science and Innovation in Society. Wiley, Hoboken (2013)

9. Shishkov, B., Larsen, J.B., Warnier, M., Janssen, M.: Three categories of context-aware systems. In: Shishkov, B. (ed.) BMSD 2018. LNBIP, vol. 319, pp. 185–202. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94214-8_12
10. Hopkins, P.: Fundamentals of Risk Management - Understanding, Evaluating, and Implementing Effective Risk Management. IRM (2012)
11. Ali, H.O., Rozan, M.Z.A., Sharif, A.M.: Identifying challenges of change impact analysis for software projects. In: Proceedings of International Conference on Innovation Management and Technology Research, Malacca, Malaysia. IEEE (2012)
12. Wegdam, M.: AWARENESS: a project on context AWARE mobile NEtworks and ServiceS. In: Proceedings of 14th Mobile & Wireless Communications Summit. EURASIP (2005)
13. Shishkov, B., Hristozov, S., Verbraeck, A.: Improving resilience using drones for effective monitoring after disruptive events. In: Proceedings of 9th International Conference on Telecommunications and Remote Sensing (ICTRS 2020). Association for Computing Machinery, New York (2020)
14. Silvander, J.: On context frames and their implementations. In: Shishkov, B. (ed.) Business Modeling and Software Design. BMSD 2021. LNBIP, vol. 422. Springer, Cham (2021)
15. Shishkov, B.: Tuning the behavior of context-aware applications. In: Shishkov, B. (ed.) Business Modeling and Software Design BMSD 2019. LNBIP, vol 356, pp. 134–152. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24854-3_9
16. Shishkov, B.: Designing Enterprise Information Systems Merging Enterprise Modeling and Software Specification. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-22441-7
17. Bunge, M.A.: Treatise on Basic Philosophy. A World of Systems, vol. 4. D. Reidel Publishing Company, Dordrecht (1979)
18. Shishkov, B., Mendling, J.: Business process variability and public values. In: Shishkov, B. (ed.) BMSD 2018. LNBIP, vol 319, pp. 401–411. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94214-8_31
19. Dey, A.K., Newberger, A.: Support for context-aware intelligibility and control. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems. ACM, USA (2009)
20. Bosems, S., van Sinderen, M.: Models in the design of context-aware well-being applications. In: Meersman, R., et al. (eds.) OTM 2014. LNCS, vol. 8842, pp. 37–42. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45550-0_6
21. Cano, J., Delaval, G., Rutten, E.: Coordination of ECA rules by verification and control. In: Kühn, E., Pugliese, R. (eds.) COORDINATION 2014. LNCS, vol. 8459, pp. 33–48. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43376-8_3
22. Alegre, U., Augusto, J.C., Clark, T.: Engineering context-aware systems and applications. J. Syst. Softw. **117**, 55–83 (2016)
23. Alférez, G.H., Pelechano, V.: Context-aware autonomous web services in software product lines. In: Proceedings of 15th International SPLC Conference. IEEE, CA, USA (2011)
24. Abeywickrama, D.B., Ramakrishnan, S.: Context-aware services engineering: models, transformations, and verification. ACM Trans. Internet Technol. J. **11**(3), Article 10 (2011)
25. Borissova, D., Cvetkova, P., Garvanov, I., Garvanova, M.: A framework of business intelligence system for decision making in efficiency management. In: Saeed, K., Dvorský, J. (eds.) CISIM 2020. LNCS, vol. 12133, pp. 111–121. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-47679-3_10
26. Garvanova, M., Garvanov, I., Kashukeev, I.: Business processes and the safety of stakeholders: considering the electromagnetic pollution. In: Shishkov, B. (ed.) BMSD 2020. LNBIP, vol. 391, pp. 386–393. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52306-0_28

27. Dimitrova, Z., Dimitrov, V., Borissova, D., Garvanov, I., Garvanova, M.: Two-stage search-based approach for determination and sorting of mountain hiking routes using directed weighted multigraph. Cybern. Inf. Technol. **20**(6), 28–39 (2020). Print ISSN 1311-9702, Online ISSN 1314-4081. https://doi.org/10.2478/cait-2020-0058
28. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: First Workshop on Mobile Computing Systems and Applications, pp. 85–90. IEEE (1994)
29. Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. Wirel. Netw. **8**, 187–197 (2002)
30. Dey, A.K.: Context-aware computing: the CyberDesk project. In: AAAI Spring Symposium on Intelligent Environments, AAAI Technical Report SS-88-02, pp. 51–54 (1998)
31. Abecker, A., Bernardi, A., Hinkelmann, K., et al.: Context-aware, proactive delivery of task-specific information: the knowmore project. Inf. Syst. Front. **2**, 253–276 (2000)
32. van Sinderen, M., van Halteren, A., Wegdam, M., et al.: Supporting context-aware mobile applications: an infrastructure approach. IEEE Commun. Mag. **44**(9), 96–104 (2006)
33. Chaari, T., Laforest, F., Celentano, A.: Adaptation in context-aware pervasive information systems: the SECAS project. Int. J. Perv. Comput. Commun. **3**(4), 400–425 (2007)
34. Kegel, R.H.P., van Sinderen, M., Wieringa, R.J.: Towards more individualized interfaces: automating the assessment of computer literacy. In Proceedings of 7th International Workshop on Behavior Change Support Systems, BCSS@PERSUASIVE 2019, CEUR Workshop Proceedings, vol. 2340. CEUR-WS.org (2019)
35. Sun, D., Paredes, P., Canny, J.: MouStress: detecting stress from mouse motion. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York (2014)
36. Pawar, P., Van Beijnum, B., Hermens, H., Konstantas, D.: Analysis of context-aware network selection schemes for power savings. In Proceedings of Asia-Pacific Services Computing Conference, pp. 587–594. IEEE (2008)
37. Van Engelenburg, S.: Designing context-aware architectures for business-to-government information sharing. Ph.D. thesis. TU Delft Press (2019)