Integrating Privacy During Requirements Capture for Ubiquitous Computing

Richard Gunstone
Computing and Informatics, School of Design, Engineering and Computing
Bournemouth University, Poole, Dorset, United Kingdom
rgunstone@bournemouth.ac.uk

***Abstract*—Use cases are used extensively in software requirements capture and representation in contemporary software system analysis and design. Ubiquitous computing systems are likely to require a high degree of user-centric requirements capture if they are to meet the often demanding requirements of the modern computer user. Such systems have a strong association with privacy issues, and their users are likely to have highly adaptive, complex privacy needs. This paper proposes a synthesis and extension of exant approaches to misuse cases for the special case of privacy issues in developing ubiquitous computing systems, catering for privacy enhancing technology. By incorporating privacy at the earliest and most critical phase of system development, we propose the enhancement provides greater emphasis on addressing privacy needs during systems development.**

***Index Terms*—use cases; privacy; ubiquitous computing**

I. Introduction

Use cases have found frequent use in software requirements capture and representation in contemporary software engineering and they are frequently integrated into business processes. The requirements engineering process is widely recognised as being crucial in the process of building a software system, by building a specification through iterative processes of elicitation, specification, and validation, ideally integrating multiple viewpoints to encourage objectivity [1].

The concept of *ubiquitous computing* is often attributed to Mark Weiser [2]. His vision of a pervasive, connected world through which computing would adapt to the environment, rather than the reverse, holds significant promise as a future paradigm for computing. Similarly, we view ubiquitous computing as a move toward an environment where technology diffuses into the background and where software systems are used that adapt to user needs autonomously. Ubiquitous computing examples are wide and varied, but they tend to require relatively advanced functionality in networks, context-awareness, and interaction design. Recent developments in consumer technology, particularly the convergence of technologies and information on mobile devices, have highlighted a growing interest in computers serving as communication tools in societal contexts. Use cases offer the potential to support the development of ubiquitous computing technology, where a high degree of user-centric requirements capture is likely to be necessary.

Use cases (or scenarios) serve as projections of future system usage and as projected visions of interactions with a designed system [3], providing a number of benefits to both requirements and software engineering. Use cases–as proposed originally and referred hereafter to as Ordinary Use Cases or (OUCs)–offer scalability and iterative improvement support [1], an intuitive graphical notation scheme; their use of natural language offers benefits such as partial specification and an opportunity to capture the user viewpoint [4]. They are also thought to suffer a number of disadvantages, including a lack of formal syntax and semantics that can lead to ambiguity, and not being amenable to formal analysis [1, 5].

While use cases offer benefits, their limitations have led to several proposals that aim to extend or replace the core modelling technique. In the following parts of the paper we focus particularly on privacy extensions to the requirements capture part of the development process. These extensions sit within a broader effort to address many of the problems identified with the original use case approach adopted by practitioners.

Modifications proposed in the literature sit within a spectrum of primarily augmentation approaches (i.e. incremental enhancements) on the one hand, and more extensive replacements formalisms on the other. Examples include incorporating goals and Non-Functional Requirements (NFRs), [6], e.g., the work of Lee and Xue [5], the use of more structured representations [4], using a more formal methodology for pre- and post-conditions [7], or considering the system within the environment [8]. Examples of more radical changes to the use case representation include the use of Petri nets via Constraints-Based Modular Petri Nets (CMPNs) [1] and Place Transition nets (PT or PrT nets) [9] and the generation of test requirements. We recognise some of these formalisms (e.g., PrT nets) are particularly interesting in their suitability for representing cross-cutting requirements such as security requirements.

We do make the implicit assumption however that augmentation of the methodology is appropriate for the purposes of privacy, rather than a complete replacement of the formalism. We note the observation made by Glinz, that while formal representations are useful for analysis and can achieve high levels of precision, this comes at the expense of readability and effort to write the scenarios [4].

The remainder of this paper is structured as follows. In section II, we explore the nature of privacy in the context of ubiquitous computing, contrasting to the similar, sometimes overlapping but distinct concept of information security. In section III, we attempt a synthesis of extant approaches to MUCs (that cater primarily for information security) and adapt these to the special case of privacy issues in developing UCSs. The paper is finished with conclusions and acknowledgements.

II. A NEED TO CATER FOR PRIVACY

We make the distinction in this paper between on the one hand *information security*, which is primarily concerned with the confidentiality, integrity, and availability of information, and *privacy* on the other. While Information Security is often an important element in ensuring effective privacy controls, it does not encompass all aspects of privacy provision. Similarly, privacy is not equivalent to information security.

Privacy is a difficult concept to formally specify, particularly for ubiquitous computing (and for computing more generally). At a theoretical level, theorists have evolved their articulation of privacy since the early legalistic definitions [10]. There has been a particular change in immediacy from the likes of *territorial privacy* toward more contemporary notions of *remote privacy* concerned with communications and information. As part of this change, privacy has evolved into a multidimensional concept (c.f. Marx's *personal border crossings* for *natural*, *social*, *spatial/temporal*, and *ephemeral/transitory* aspects of privacy [11]).

On a practical level moving from theories of privacy, which are useful in exploring the efficacy of the concept, toward building complex ubiquitous computing systems (UCS), inevitably requires specific technical developments to the system design to ensure privacy can be protected or enhanced. This necessitates the implementation of technical measures–*Privacy Enhancing Technology (PET)*. (Indeed, the introduction of PET serves to contrast ubiquitous computing privacy versus the classical models of privacy mentioned earlier.)

Individual users typically have dynamic privacy expectations. Such expectations vary according a multiplicity of factors (that are much more specialised than the broad themes proposed by Marx), for example: the other parties involved, timing, events, decisions recently made, the type of information to be shared, what the information will be used for. In tackling this highly variable user requirement, the provision of a suite of technical measures to fortify privacy and controlled by the user, therefore becomes crucial. (Indeed, this trend to push the locus of control for individual privacy to the user is increasingly commonplace in other areas of computing e.g., social networks).

Ubiquitous computing takes privacy considerations for contemporary computing much further, because a UCS is intrinsically based upon the collection and processing of information about their users, the environment, their property, actions, and so on. This information is collected all or most of the time, senses information types that are not readily accessible in contemporary computing paradigms (such as video camera feeds of rooms, contents of fridges via RFID, etc.), and is extensively processed to yield new useful information that can aid the activities of its users. Thus we argue a special treatment of privacy in ubiquitous computing is necessary and justified.

It is worthwhile at this point elaborating on what this paper regards as PET. We define PET to be 'technical means through which privacy can be preserved or enhanced according to the needs of the user'. PET can be implemented under a variety of categories, with some typical examples including: (1) *Information management policies* and provision of policy specifications and management infrastructure to govern the release of information, to whom and for what purpose, for how long, etc.; (2) *Statistical anonymisation*, encompassing techniques such as k-anonymity, and location cloaking such as adding noise, or discretization of location data; (3) *Communication privacy* including traditional approaches to ensuring privacy (encryption); (4) *Identity authentication and authorisation*, including federated identity technologies such as Shibboleth. (The broad range of PET categories serves to underline the need to derive PET from an analysis of risk during the design and development of a UCS.)

Privacy risks in ubiquitous computing are not often readily apparent, but surface in unanticipated ways often through the use of sources of information in unanticipated ways. A typical example is a ubiquitous computing infrastructure that collects information over a period of time and then due to an absence of privacy safeguards allows this information to be made available to an external party who takes actions that are unwanted by the user. For example, a smart space (environments equipped with extensive ubiquitous computing infrastructure) that monitors the amount of waste over a period of time, but then makes this information available to a local waste authority who made a supplementary charge for the property, clearly raises privacy and ethical questions. (For example, should UCS users be at a disadvantage to other members of society who do not use UCSs?)

Fundamentally, the use of ubiquitous computing technology implies the sharing of the association between identity and types of data that are often considered personal, for instance real-time location information. Poorly designed ubiquitous computing architectures open the opportunity for information to made available, and subsequently used, in a way that does not uphold such privacy requirements.

Addressing privacy considerations is, then, a mixture of *countermeasures to enhance privacy* (sometimes referred to as *privacy enhancing technology* or *PET*) and a *consideration of potential privacy-breaching activities*. We established in Section 1 that the requirements engineering process is widely recognised as being crucial in the process of building a software system, and thus we argue the process of identifying suitable countermeasures should begin when requirements are being established. Similarly, use cases are a logical methodology to adopt given the benefits discussed earlier. However, in their extant form use cases are not ideal in that they tend to model the intent of cooperative users of the system.

We draw attention to this, and underline the need to be able to explicitly represent (1) PET; (2) the uncooperative users of the system (or information) and (3) the relationships between OUCs, privacy risks and PET. While it may be possible to use OUCs to represent privacy requirements, without explicit elements within the formalism to represent these aspects the requirements engineering process is potentially at risk of losing information. Crucially, such information would be useful for subsequent phases of UCS development.

As has been established, the existing meta-model for OUCs does not provide formalism for representing these extensions. Thus, it is desirable to extend the use case methodology to accommodate privacy concerns, and to potentially have a means to develop the requirements and consequential system design in such a way as to address any concerns identified during use case analysis. Any PET identified can then be developed sourcing appropriate requirements in the use case model.

III. Representing Privacy using Use Cases

In developing a use case augmentation that can cater for privacy, there are several considerations. Firstly a representation must be able to represent the risks to privacy; that is, the representation should explicitly model the risk formally without recourse to supplementary models. The representation should also be able to represent the PET to counter such risks; it should contain sufficient information to lead to the development of PET as requirements are carried through from the requirements capture phase through all subsequent phases of the ubiquitous computing project. Finally, the augmentation should provide a straightforward way of translating all relevant use cases into implementation, particularly with respect to privacy features.

One approach to representing privacy concerns is to extend the use case diagram to accommodate features that are complementary to the 'cooperative' elements discussed earlier. A class of use case formalism, referred variously as *misuse cases* and *abuse cases*, has emerged in recent years, primarily focused on Information Security risk analysis but we argue applicable more broadly to privacy.

Alexander presents a well-known extension using the "misuse case" (MUC) [12], that has been acknowledged in various studies. This concept simply represents a negative scenario, a use case from the point of view of an actor hostile to the system. An example of a malevolent actor might be a car thief, a hacker, a rogue employee, or non-human entity (or process such as the weather). Such actors are represented diagrammatically as an additional actor. Using Jacobson's original use case formalism, the MUC is represented as a series of parallel use cases that threatens the ability of the friendly actor to perform an ordinary use case. A MUC describes potential system behaviours that are deemed unacceptable to the stakeholders of a system [13]. MUCs are structured and use the same representational principles as ordinary use cases. Their connection to ordinary use cases are represented via *«threatens»* and *«mitigates»* relationships between the two types of case respectively, leading to a zigzag pattern of play and counter-play. In so doing, they potentially represent a means through which risk can be explicitly modelled.

In Alexander's approach, an iterative style of development leads to an enumeration of MUCs and additional use cases that can mitigate the threat posed by MUCs. During the requirements elicitation process the requirements analyst should enumerate each stake holder's 'doomsday scenario', in order to identify potential misuse cases [13]. Through an iterative process it should be feasible to enumerate all potential use cases and MUCs (as far as practicable). MUCs also provide a basis for developing new subsystems and components that lead to countermeasures. These developments may themselves lead to new types of threat, that require the evaluation of further countermeasures, thus spurring on the iterative process to conclusion.

In practice countermeasures derived from use case analysis are likely to vary depending on context of use. For instance, a financial trading system may use the same software product family as a small business, however the consequences should confidentiality be compromised are potentially more significant. Authentication in such a case, may require a more complicated form of authentication rather than simple password authentication. (It follows, therefore, that some variance in implementation may emerge depending on the software product line.)

This particular aspect of uses cases is explored in [14] where the authors propose the introduction of separate use cases to the use case diagram (i.e. the development of a parallel model to support the main use case model). Their rationale for separation is to reflect the variability in application and security requirements and consequently they propose a separation of security concerns is appropriate for use cases. In this approach, Security Use Cases (SUCs) are specified separately to the application use cases and can incorporate parametrisation. SUCs are subsequently integrated into the use case diagram using *extension points*, which focus as the "bridge" between the UC and SUC models. For instance, a parametrised SUC may be defined for an authentication service, which determines whether the accessor is a valid user. This could be subsequently fulfilled by two alternate use cases for example, one authenticating using an ID and password, and the other authenticating using a biometric (such as a fingerprint). This leaves the application use cases to specify business functionality, and the parameters for the SUCs are set, as mentioned, on the basis of the product line.

Similarly, Rosado et al. [15] (looking at Grid computing) also identify a need for SUCs, and propose extensions including specialisations of the generic use case for misuse, mobility, and security (Røstad proposes an extended notation for misuse cases [16] particularly focused on Information Security and a review of previous work in the area of misuse cases).

A number of observations are appropriate on the basis of such studies, particularly with a view to the development of PET in ubiquitous computing: (1) it is not feasible at the requirements capture stage to overlook events that may impact the privacy expectations of the anticipated users of a system. As the introduction noted, the requirements engineering process is widely recognised as being crucial in the process of building a software system [1]. Indeed retro-fitting PET to a system that has not been developed from first-principles to cater for privacy is unlikely to be economic or straightforward; (2) use case models need to represent additional actors in addition to the anticipated actors or users of the system. At a minimum an additional class to cater for "misusers" or similar of a system is necessary; (3) a commonality across several extensions to the use case formalism is that a further type of use case is necessary to cater for categories of risk that misusers of a system present (the "misuse case"); (4) a further use case type is necessary to encapsulate countermeasures; and, (5) extension points offer a means through which a use case model can 'drill down' to a level of detail suitable for the development of countermeasures.

We thus envisage two actors as follows: (1) The uncooperative user or actor (akin to "misuser" in MUCs) is an individual who wishes to invade the privacy of a legitimate user of the system; (2) The legitimate user or actor ("ordinary actor") is one who the UCS is designed for and both performs legitimate actions using the UCS, and performs actions that protect or enhance his or her own privacy. Considered together, three tiers of activity emerge, leading to the following types of use case ('prototypes' in the terminology of Rosado et al.):

OrdinaryUseCase (UC)–this is a use case of the form currently used in mainstream use case modelling, representing the legitimate actions of the "ordinary actor". This continues to represent a flow of events [1], includes a description comprising structured or unstructured text [17], contains a sequence of actions (or transactions) performed by the system leading to an observable result that is of value to an actor [17, 18, 19]. The standard relationships *«extend»* and *«include»* are compatible with this type. UCs are evolved using standard, widely-accepted practices.

PrivacyMisuseUseCase (PMUC)–this represents a privacy risk to the privacy expectations of ordinary users of the system. Within this context, the PMUC represents the characteristics associated with the standard use case. The relationships *«threaten»* and *«mitigate»* are compatible with this type, after Alexander [12]. (To aid reproduction, we represent the PMUC as a double-circled variant of the standard use case.)

PrivacyEnhancingUseCase (PEUC)–this represents a *privacy-enhancing use case*, and acts as the anchor point for PET. We propose this type will normally be associated with an extension point that permits a range of PET options. The relationship *«threaten»* (from a PMUC) is compatible with this type. (This diverges from Røstad's proposal, which uses the standard use case type for countermeasures and is more straightforward.)

PETUseCase (PET)–anchors to the extension point of the PEUC and can use the *«mitigate»* relationship. This is used to represent PET within the model. At the extension point we propose they may or may not be included as part of the main use case model, decided as necessary by the analyst. (Diverging from the practice advocated in [14].) The relationship *«mitigate»* is compatible with this type, after Alexander [12].

Further, we define the following additional type of actor:

UncooperativeActor (UA)–an actor associated with PMUCs, for whom the successful completion of their nefarious activity achieves value. While the use case model is not intended to cater for this type of actor in terms of services offered by the system, their inclusion is (as the paper implies) to develop countermeasures to safeguard the privacy of the intended users. Within this context, the UA shares the properties and characteristics of a standard actor in the use case meta model. (This actor is complemented by the conventional UML actor, which we abbreviate as Cooperative Actor or CA.)

A complexity with this proposal when representing privacy at the use case analysis stage is how best to represent the interplay between privacy threats and the actions of the cooperative user of the system. Encapsulating this by linking PMUCs directly to OUCs (via e.g., <<*threaten*>>) might appear sufficient, however this implies a threat to the actions of the CA more akin to traditional Misuse Cases (MUCs), whereas in the case of privacy, the CA will usually be able to perform their actions regardless of any privacy threats, with privacy risks surfacing subsequently and not directly jeopardising the original activity. The unwanted tracking of a UCS user, for instance, does not necessarily jeopardise the sharing of location information. We propose this logical impasse can be overcome by assuming the areas where the CA wishes to enhance privacy are known to the CA and these are correspondingly represented separately in a second middle tier; the interplay then occurs between the PMUCs of the UA and the PEUCs (and PETs) of the CA. (This leaves the conventional use case analysis separate.) Figure 1 shows these new constructs diagrammatically.

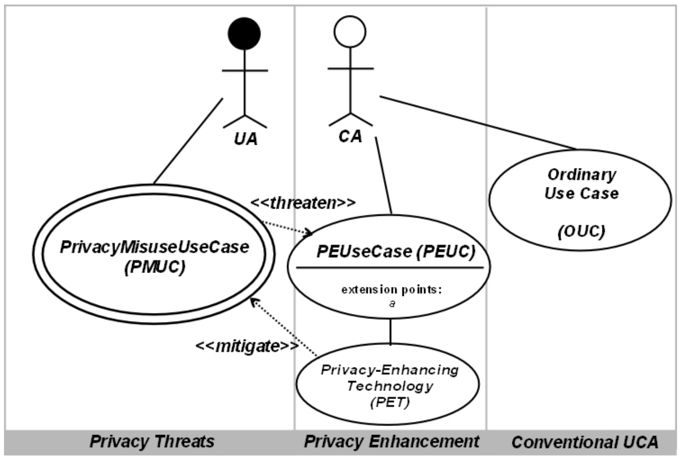

Figure 1. The main diagrammatic elements for modelling privacy in the proposed model. The introduction of a third tier concerned with *Privacy Enhancement* serves to contrast the proposed approach with traditional MisuseCases (MUCs).

This additional group of PEUCs can be enumerated according to the following outline repetitive process for developing and elaborating the use case model: (1) Identify, enumerate and represent the Ordinary Use Cases (OUCs) of the CA; (2) Identify, enumerate and represent the Privacy Misuse Use Cases (PMUCs) of the UA. (These represent the privacy threats to the normal performance of OUCs.); (3) Identify, enumerate and represent Privacy Enhancing Use Cases (PEUCs) to counter the privacy threats posed by PMUCs. Generic PEUCs are appropriate in this type of use case, for instance "Cloak location data"; (4) Identify, enumerate and represent Privacy Enhancing Technology (PET) use cases attached to relevant extension points on the PEUCs identified above. An example of PET for an extension point for the previous example may be "Discretize location" or "Implement k-anonymity or return no information"; *(5) Repeat steps until use case analysis is complete.*

In terms of their textual representation, PMUCs, OUCs, and PEUCs should share the conventional textual representation of contemporary use cases, adapted as appropriate to privacy.

IV. Conclusion

Use cases have achieved widespread use in software engineering problems, and more widely as a generic term for user needs from products and services.

In this work-in-progress paper, we have proposed a synthesis of extant approaches to MUCs and evolved these to the special case of privacy issues in developing UCSs. The proposed approach is particularly attuned to the PET requirements of ubiquitous computing.

In modelling privacy at the use case development stage a number of advantages are evident:

- Privacy is explicitly modelled early-on during the system development process (the most critical phase of system development). This provides an opportunity to carry through the privacy requirements identified during use case analysis through into system development.
- Privacy-enhancing technology is accommodated in the use case model, and the analyst has a choice as to whether to include PET use cases directly via extension points into the use case model, or to represent these separately.
- The use case development process for misuse cases provides an outline of the iterative process required to develop use cases, PET and associated privacy-related use cases. Through iterative development, PET should emerge from the use case model to enhance the system development process.
- The modifications represent an augmentation to the standard use case methodology, providing familiarity and a straightforward development process for software engineering and development.

Further developments to this approach include validating the methodology against a real-world system example, and introducing a more formalised description of the elements introduced (akin to the modelling approached used in [14]).

V. Acknowledgements

The author is particularly grateful to the anonymous reviewers for their fair and reasoned contributions.

References

[1] W. J. Lee, S. D. Cha, and Y. R. Kwon, "Integration and Analysis of Use Cases Using Modular Petri Nets in Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 24, pp. 1115–1130, December 1998.

[2] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," *Commun. ACM*, vol. 36, pp. 75–84, July 1993.

[3] A. G. Sutcliffe, N. A. M. Maiden, S. Minocha, and D. Manuel, "Supporting Scenario-Based Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 24, pp. 1072–1088, December 1998.

[4] M. Glinz, "Improving the Quality of Requirements with Scenarios," in *Proceedings of the Second World Congress for Software Quality (2WCSQ)*, (Yokohama), pp. 55–60, September 2000.

[5] J. Lee and N.-L. Xue, "Analyzing User Requirements by Use Cases: A Goal-Driven Approach," *IEEE Software*, pp. 92–101, July/August 1999.

[6] L. M. Cysneiros and J. C. S. do Prado Leite, "Non-functional Requirements: From Elicitation to Conceptual Models," *IEEE Transactions on Software Engineering*, vol. 30, pp. 328–350, May 2004.

[7] C. Nebut, F. Fleurey, Y. L. Traon, and J.-M. Jezequel, "Automatic Test Generation: A Use Case Driven Approach," *IEEE Transactions on Software Engineering*, vol. 32, pp. 140–155, March 2006.

[8] J. B. Jorgensen and C. Bossen, "Executable Use Cases: Requirements for a Pervasive Health Care System," *IEEE Software*, pp. 34–41, March/April 2004.

[9] D. Xu and X. He, "Generation of Test Requirements from Aspectual Use Cases," in *Proceedings of WTAOP07 Workshop*, (Vancouver, British Columbia, Canada), pp. 17–22, ACM, March 2007.

[10] S. D. Warren and L. D. Brandeis, "The Right to Privacy," *Harvard Law Review*, vol. IV, December 1890.

[11] G. T. Marx, "Murky Conceptual Waters: The Public and the Private," *Ethics and Information Technology*, vol. 3, no. 3, pp. 157–169, 2001.

[12] I. Alexander, "Misuse Cases: Use Cases with Hostile Intent," *IEEE Software*, pp. 58–66, January/February 2003.

[13] G. Peterson and J. Steven, "Defining Misuse within the Development Process," *IEEE Security and Privacy*, pp. 81–84, November/December 2006.

[14] H. Gomaa and M. E. Shin, "Separating Application and Security Concerns in Use Case Models," in *Proceedings of EA09*, (Charlottesville, Virginia, USA), ACM, March 2009.

[15] D. G. Rosado, E. Fernandez-Medina, and J. Lopez, "Reusable Security Use Cases for Mobile Grid Environments," in *Proceedings of ICSE09 Workshop, SESS09*, (Vancouver, Canada), pp. 1–8, IEEE, May 2009.

[16] L. Rostad, "An extended misuse case notation: Including vulnerabilities and the insider threat," in *Twelfth Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ06)*, (Luxembourg), June 2006.

[17] D. Leffingwell and D. Widrig, *Managing Software Requirements: A Use Case Approach*. Addison-Wesley Professional, 2 ed., May 2003. Print ISBN-10: 0-321-12247-X; Print ISBN-13: 978-0-321-12247-6.

[18] I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse: Architecture, Process and Organisation for Business Success*. Reading, MA: Addison-Wesley/ACM Press, 1997.

[19] A. J. H. Simons, "Use Cases Considered Harmful," tech. rep., University of Sheffield, 1999.