Empowering users to control their privacy in context-aware systems through interactive consent

Maarten Wegdam

Telematica Instituut
P.O. Box 589, 7500 AN
Enschede, The Netherlands
maarten.wegdam@telin.nl

University of Twente
Department of Computer Science / CTIT
P.O. Box 217, 7500 AE
Enschede, The Netherlands
wegdam@cs.utwente.nl

Dirk-Jaap Plas

Alcatel-Lucent
Larenseweg 50, 1221 CN, Hilversum
dplas@alcatel-lucent.com

ABSTRACT

Context-aware systems adapt their behaviour based on the context a user is in. Since context is potentially privacy sensitive information, users should be empowered to control how much of their context they are willing to share, under what conditions and for what purpose. We propose an interactive consent mechanism that allows this. It is interactive in the sense that users are asked for consent when a request for their context information is received. Our interactive consent mechanism complements a more traditional pre-configuration approach. We describe the architecture, the implementation of our interactive consent mechanism and a use case.

Author Keywords

Context, privacy, interactive consent, context-aware applications

INTRODUCTION

Context information, or context for short, describes the situation a user is in. Examples are where someone is and what he is doing, e.g., whether someone is working or not. Context-awareness is an essential characteristic of the pervasive, ubiquitous computing environment that Mark Weiser envisioned [20] and that is gradually becoming a reality due to improved sensor technologies, improved mobile network coverage, smaller computing devices, etc. Context is sensed by physical sensors (e.g., a GPS receiver, that provides location information) or by a logical sensors (e.g., a software component that reads someone calendar), and can be combined and reasoned with to produce higher-level context (e.g., is someone available to receive a phone call?). Context information is potentially privacy sensitive information, and even apart from legal obligations, for users to accept context-aware applications they need to feel in control over this privacy [8].

P

The existing practice for applications to obtain consent to gather and use personal information is to let the user accept a legal statement that is incomprehensible for the average user, also known as click-through agreement. The user only has the choice to consent to this agreement or to not use the services at all (takes all / leave all). We believe that this is approach is not suitable to be applied in ubiquitous computing environments, and does not provide actual privacy control for the user. With the introduction and uptake of context-aware applications the amount of personal information that is collected is increasing, as well as the number of applications that make use of this information. To be able to protect ones privacy people need to know what context is collected about them and how it is used, and they need to have means to control this. Therefore, users should be provided with an actual and personalized privacy control, in which they can control what context to share in a fine-grained manner. This will increase user acceptance, and provide a value-add over take all/ leave all type of consent.

In cases where a more fine-grained privacy control is offered, this is usually based on pre-configuration: users have to configure in advance which people or applications can have access to particular information. A major issue when providing users with a fine-grained control over their privacy is that this quickly will become too complicated for most users to understand within the amount of time they are willing to spent to configure their privacy policies. In addition, due to the dynamicity of the pervasive environments that context-aware systems will execute in, users do not know which service they will use, and in what context they will use them in. They can therefore not be expected to configure their privacy preferences in advance.

This paper proposes to use an interactive consent approach in which users are asked for their consent at the point in time some application or other user requests their context. They can then, at that point in time and thus given the context they are in, determine if they want to share this context with the requester. The obvious risk is that the users will find the consent questions too interruptive. We therefore made it possible for users to generalize their responses to also apply to a range of future consent requests, allowing the privacy enforcement system to incrementally learn their privacy policies. The interactive consent approach we propose in this paper is complementary to more traditional pre-configuration approaches.

Besides the interactive consent mechanism itself, a second contribution of the paper is a proof-of-concept prototype of this mechanism using mobile phones. This prototype is implemented as part of a context infrastructure for context-aware mobile applications. We describe a use case where we used this interactive consent prototype for a context-aware meeting scheduling application.

This paper is structured as follows. We first describe related work. Then, we analyze the issues with providing informed user consent, and propose our interactive consent mechanism. Next, we describe the architecture of our proof-of-concept, followed by a use-case for our proof-of-concept prototype. We end this paper with conclusions and discuss possibilities for future work.

RELATED WORK

The need for more insight and control on sharing of personal information, such as context information, in ubiquitous computing like environments, is confirmed by Petterson et. al [17]. This paper maps legal privacy principles to human-computer interface (HCI) requirements. It proposes user interfaces for the user to simplify the process of obtaining consent and for the application to follow legal privacy principles and adhere to privacy legislation.

The work of Petterson et al. is based on the research conducted by the PISA project. The PISA project investigated user agreements for obtaining informed user consent. Patrick et al. describe their work in the PISA project [15]. They acknowledge that the legal statements provided by applications are often long, complex and difficult for users to read and/or understand. They therefore introduce the concept of Just-In-Time-Click-Through Agreements (JITCTA). The main feature of JITCTA is not to provide a large, complete list of service terms but instead to confirm understanding or consent on an as-needed basis.

While the work described by Petterson et al. and Patrick et al. focuses on how to convey legal privacy statement following privacy principles, our work focuses at how personal information can best be protected in a context-aware environment. Patrick et al. assume that consent only needs to be requested when users are using applications themselves and that they are providing the personal information themselves, while in context-aware environments context information is constantly being collected. This requires another way to protect this information.

One commonly applied way for users to indicate to what extend they want to expose their personal information is by pre-configuring their privacy preferences or policies. However, there are a number of drawbacks with this approach, especially in context-aware and ubiquitous computing environments. Lederer et al. state that users do not like to spend much time on specifying their privacy settings [10]. When they start using an application they are willing to configure their settings – although it should not be too cumbersome - but they do not like to do this on a regular basis. People therefore easily forget what they have configured, resulting in outdated privacy policies with possibly unwanted consequences.

A study by Beckwidth [2] shows that people not only forget what they have configured but also the fact that they are being monitored. People should be reminded to make sure that they stay aware that information on them is collected and about the consequences.

Cornwell et al. describe that people are unable to achieve high levels of accuracy when specifying their privacy preferences. Rules they had specified at the beginning of the experiment only captured their preferences 59% of the time. After revision, this number only went up to 65%. [3]. A possible reason for this is brought up by Nodder [12] who found that people cannot image the situations well enough when configuring their privacy policies, resulting in inconsistent observed behaviour. Another explanation is provided by Palen and Dourish, who state that privacy is an ongoing ‘boundary definition process’ [13]. Also Patil et al. [14] state that circumstances may change such that permissions defined yesterday are no longer applicable today. They are therefore in favour of increased system transparency to build trust. They argue that real-time feedback with a visual component may have a more significant impact on the awareness of people. Appropriate feedback mechanisms and interfaces need to be explored to further help users visualize their permission settings.

There are several papers that describe guidelines on privacy for context-aware systems. These guidelines provide requirements for our interactive consent solutions, but also illustrate the shortcomings of other solutions existing today. Lederer et al. describe a number of pitfalls when designing interactive systems with personal privacy implications based on the results of a user study [10]. The ground for these pitfalls is that people should be able to understand what information is disclosed about them and the implications to one’s privacy, and they should be able to configure their privacy settings using methods in a natural extension of the current design of these systems and the way people are used to handle them.

A paper by Langheinrich et al. [9] describes a number of guidelines based on a set of fair information practices as common in most privacy legislation in use today. These include amongst others:

- Notice – people should be notified when personal information on them is collected, stored and further processed or exchanged,
- Choice and consent – people need to have the choice whether to allow this and provide their explicit consent, and
- Access and recourse – people should have access to what information is collected on them and what has been done with this data and have means to change or remove this personal information.

In a short paper Grudin et al. [6] provides an overview of different approaches for access control to context information, amongst which the interactive access control. The approach they describe resembles the interactive consent mechanisms we propose. However the paper does not provide much detail, and especially no design or prototype details on how to realize interactive access control.

Related work by our own group [11] describes a small user study to understand how users deal with privacy-sensitive information and if informed consent can be improved by an interactive approach. The (preliminary) conclusion is that users consider interactive consent supplemental to a pre-configuration approach, not replacing this. Another conclusion was that users want their privacy policies to be people or group based, contrary to context or activity based.

INTERACTIVE CONSENT

Inherent choice between privacy and context

There are Privacy Enhancing Technologies, especially in the area of anonymisation (e.g., [5]) that can and should be used to make context-aware applications less privacy sensitive. However, for many applications users Privacy Enhanding Technologies cannot prevent that a choice needs to be made between a user’s privacy or making an application context-aware. Categories of applications for which this applies are applications in which users are simply not anonymous, and social networking type of applications. A concrete example is a mobile health application for tele-monitoring of patient with epileptic seizures. If there is a seizure, the location of the patient needs to be shared with available and nearby caregivers. This application is useless without sharing the context (location, availability) in a non-anonymized manner.

Personalized privacy by empowering users

From literature we known that the willingness of people to give up some of their privacy depends on (i) how much someone trusts the receiver of the privacy sensitive information, (ii) the usefulness of the application and (iii) the privacy sensitivity of the information [1]. These three factors are very personal – what is useful to one person is not useful for another etc. Privacy policies should therefore be personalized. This means each user need to be empowered to control the privacy vs. context trade-off. When possible this should not be a take-all/leave-all policy, but a more fine-grained policy in which users can give consent to share part of their context information.

Personalized privacy is context-aware

The willingness of users to share context information is not only personal, but also depends on the context of the user. In fact, both the sensitivity of the context information as well as the usefulness of the context-aware application depends on the user context. For example, in case of an application in which context is shared between colleagues to facilitate contact, users may only willing to share their context (location, activity) with colleagues when their own context is “working” since otherwise (i) their location may reveal a private activity they do not want to share (privacy sensitivity of the information), and (ii) since they do not want to be contacted anyway by their colleagues when not working (thus no usefulness).

Privacy management trade-off triangle

We distinguish three contradicting requirements in the development of our privacy management solution for supporting informed consent (Figure 1).

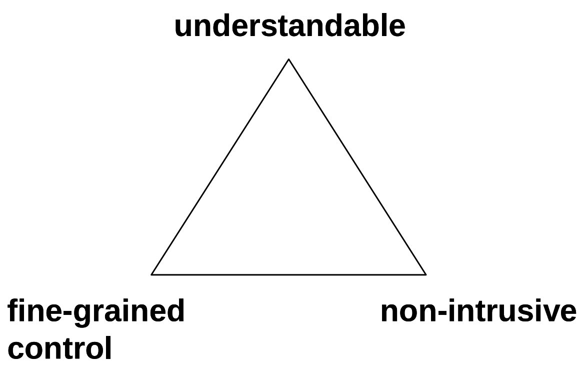


Figure 1 - The privacy management trade-off triangle

The solution should be *understandable*; users must understand what they give consent to, or put differently, the consent should be an informed consent. Users must be able to review their privacy policies, the access log of their personal information (respectively called potential and actual information flow in [10]) and be able to change their privacy policies. The solution should provide *fine-grained control*; users must be able to specify in detail under which conditions what context information may be released to who. Finally, the solution should be *non-intrusive*; users should not perceive it as annoying or interruptive by bothering the user too often, especially not if this is with repetitive consent questions or is done at inconvenient times. The challenge is that privacy management solutions cannot satisfy all three requirements since they are contradicting and therefore has to make a trade-off between these requirements. For example, fine-grained control adds complexity and thus reduces understandability, whereas asking for consent to share some context every time the system wishes to do so may be understandable and provide fine-grained control, but is too intrusive.

Our approach: ***interactive consent***

Summarizing the above, a solution is needed that provides users way to personalize their privacy consent for sharing context information, that solution needs to be context-aware itself, and should be a good compromise between understandability, fine-grained control and intrusiveness. The approach we propose in this paper is interactive consent, in which a user is asked for consent at the moment a request for context information is received. This means that users do not have to pre-configure their consent, which as we discussed in related work is a pitfall [10]. Pre-configuration is especially difficult due to the dynamicity of ubiquitous computing environments in which users will not know before hand which application they will use, and sometimes not even while using a context-aware application.

We discuss to what extend this approach addresses the three above contradicting requirements:

- *Understandability*: since a user is aware of the context he or she is in when the consent request is received, the user understands the privacy sensitivity of revealing this context, and can compare this with the usefulness of the application at that moment.
- *Fine-grained control*: the user can provide consent per context request, which is as fine-grained as possible.
- *Non-intrusive*: actually asking for consent every time some context request is received will very quickly be considered too intrusive for most if not all users. This is a major challenge for the interactive consent to be successful. To address this, we make it possible for users to give their consent for equal or similar context requests that may happen in the future. The simplest case is the “allow always” and “deny allows” options, but we also offer more sophisticated possibilities in which the user can provide what we call generalizations of the consent. Examples are “allow for all colleagues”, or even “allow for all colleagues while I’m working”. This allows the privacy enforcement system to gradually learn the user’s privacy preferences. This is further discussed in the Architecture & Design section.

Interactive consent vs pre-configuration

Interactive consent supplements a more traditional pre-configuration approach [11]. This is because of the potential intrusiveness, because it is not suitable for all types of applications (e.g., emergency applications) and because users will need to be able to inspect and change consent (potential information) that he or she provided that is valid beyond the immediate consent request.

Applications for which interactive consent is suitable will typically have one or more of the below characteristics:

- *Complex service* – for which users have complex privacy preferences, including services for which the user would want privacy policies that context-aware, and services that deal with social relationships (since social relationships are complex).
- *Service that will be forgotten or of which users are not aware* – for these types of service there is an increased risk that users do not or no longer realize that they gave consent, including service were other people can view a user’s context, but not vice-versa (e.g., an employer that can check the context of employees), long-term services that are only used infrequently, or hidden services (the true ubiquitous or 'pervasive' type).
- *Highly dynamic services* – so the user could not pre-provision the privacy policy since she would not know this service or situation would occur beforehand. This includes cases where the receiver of the context changes (e.g., community type of services in which people can join the community without consent of the user), or where the privacy sensitivity of the context depends on the context (e.g., some user may want to share his location, but not when visiting a certain place), or a service that is used only one-time.

ARCHITECTURE & DESIGN

We implement our interactive consent mechanism as part of a context infrastructure. A context infrastructure is a middleware layer that facilitates development of context-aware applications by offering context management, context reasoning, privacy and other generic functionalities [18].

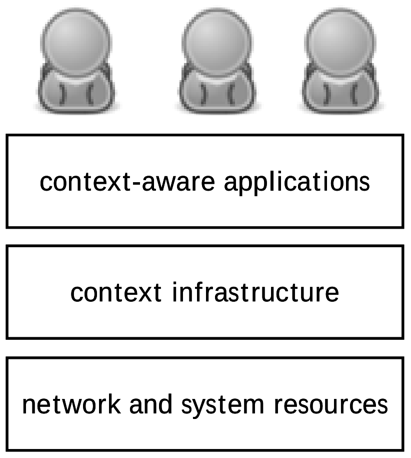


Figure 2 - Architecture of context infrastructure

Figure 2 provides a high-level layered view of such a context infrastructure. The context infrastructure collects context information from multiple context sources, and provides context-aware applications access to this information through request-response and subscribe-notify mechanisms. This context data can be raw or reasoned (i.e. aggregated or derived information). The context infrastructure provides access control based on personalized privacy policies.

The context infrastructure contains both a policy decision point as well as policy enforcement points for processing the privacy policies of the users.

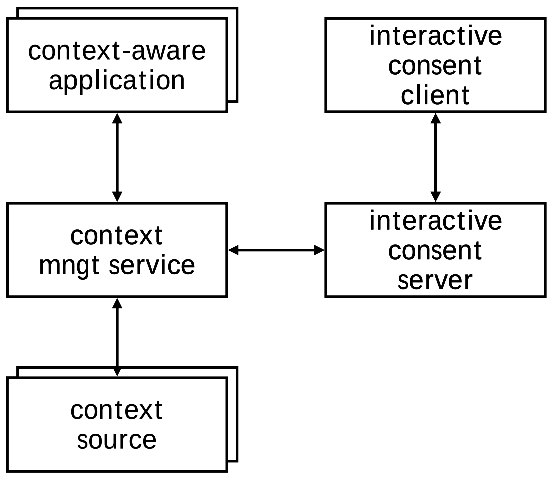

context-aware application
interactive consent client
context mngt service
interactive consent server
context source

Figure 3 - interactive consent positioned in context architecture

Figure 3 positions interactive consent in the context infrastructure. The policy decision point functionality is taken care of by the interactive consent server. When a request for context information comes in and the existing privacy policies don't provide a definite answer on whether or not to provide access, the user is asked for consent via the interactive consent client.

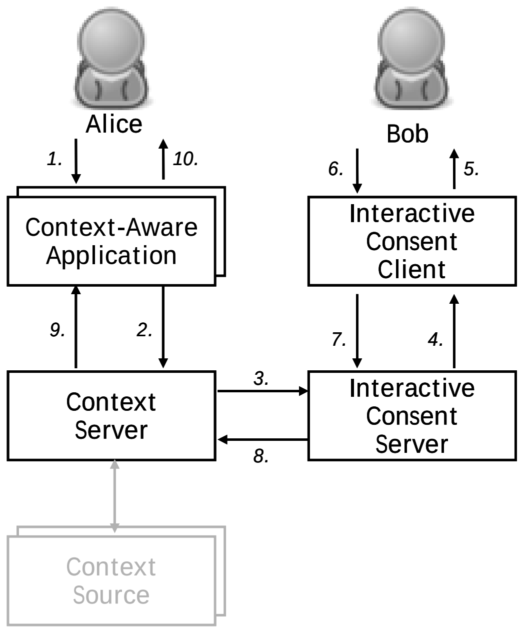

Alice
Bob
1.
10.
6.
5.
Context-Aware Application
Interactive Consent Client
9.
2.
7.
4.
3.
Context Server
Interactive Consent Server
8.
Context Source

Figure 4 - interactive consent message flow

Figure 4 illustrates all the steps that are taken in this process. User Alice uses a context-aware application (1) that requests context information of user Bob (2). The policy for accessing this information has been configured to be interactive. In this case the Context Management Service (the policy enforcement point) asks the Interactive Consent Server (the policy decision point) (3) that sends a request for interactive consent to Bob using the Interactive Consent Client running on one of Bobs devices (4,5). Bob can inspect the request and based on the information provided with the request can make a decision whether to approve, deny or ignore the request (6). The consent is passed on by the Interactive Consent Client and Server to the Context Management Service (7,8) that grants (in case of approve) or denies (in case of deny or ignore) access to the requested context information (9,10).

Main architectural choices

Above we described the main architectural components and a typically flow of interactions between them. Below we detail this by discussing how we addressed the main architectural challenges.

Interact with users via their smart phone

The interactive consent server needs to interact with user when his or her consent is required. This is done via the interactive consent client application. Since most people carry a mobile phone with them nowadays, we chose to implement the interactive consent client on a smart phone. Standard interaction mechanisms on mobile phone (typically SMS) are not sophisticated enough to implement interactive consent with, we there implemented a dedicated interactive consent client application. This client application registers with the interactive consent server and listens for incoming consent requests. When a request for consent comes in, the client application shows a notification to the user (see Figure 5). The user can respond to the consent request, which will either grant or deny the requestor access to the information he asked for. The user can also simply ignore the consent request, in which the request will time out and access to the context information will of course be denied.

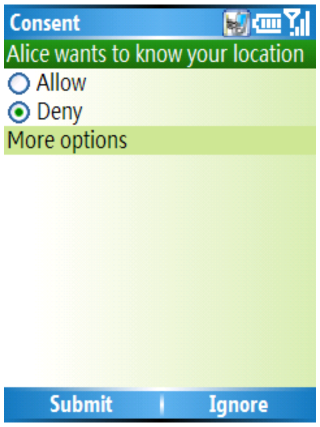

Consent
Alice wants to know your location
Allow [✓ Deny]
More options
Submit | Ignore

Figure 5 - request for consent

Some people argue that notifications should not be used and will be ignored by the user [16]. Other studies show that notifications are considered helpful by the user [7]. We follow the second strain of thought, arguing that whenever the user is presented with useful information and this does not happen too often (i.e., does not disturb the user), notifications are a good way to attract the attention of a user.

Simple questions with option for more details

As argued before, we want to offer fine-grained control but at the same time make it understandable. And we have to cater for people that want less or more fine-grained control. We chose to address this by offering a simple choice to the user, but with an option to get more details and possibilities for more fine-grained control if desired. The simple choice shows who is requesting the context, and what context information. This allows the user to decide whether he or she want to share the context based on the trust in the requester as well as on the privacy sensitivity of the information (i.e., the context requested and its value at that point in time), which as e.g., [12] has shown are important factors.

Figure 5 already showed how the user initially is only presented with a simple question: "<requester> wants to know your <context information>, where <requester> is the name of the requester and <context information> can be any kind of context delivered by the context server, such as location or availability. Figure 6 gives an example of how user is presented with more details on the request if the user selects "More options". Details include with what application the requesting user is accessing the context, whether it is a one-time request-response or a subscription, how the requesting user was authenticated and what context information would be shared. This includes the quality of the context, since higher quality information typically is more privacy sensitive [19]. An obvious example of this is location: which building someone is in is more privacy sensitive than which city.

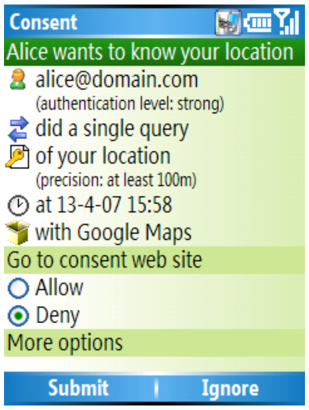
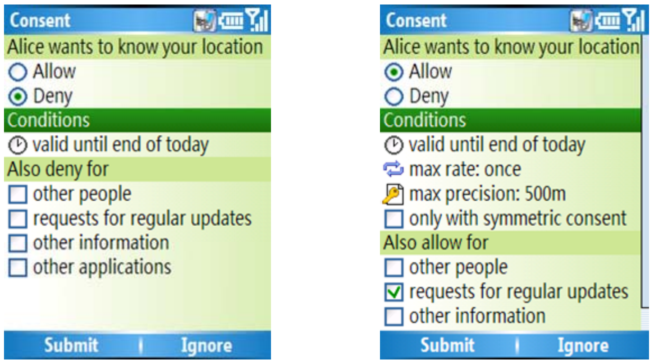

Consent
Alice wants to know your location
alice@domain.com
(authentication level: strong)
did a single query
of your location
(precision: at least 100m)
at 13-4-07 15:58
with Google Maps
Go to consent web site
Allow [✓ Deny]
More options
Submit | Ignore

Figure 6 - additional details

Generalization of the answer

To prevent that user are flooded with requests for consent each time someone enquires their context information, we have implemented a mechanism for minimizing the number of consent requests by generalizing the consent responses.


Consent
Alice wants to know your location
Allow [✓ Deny]
Conditions
valid until end of today
Also deny for
☐ other people
☐ requests for regular updates
☐ other information
☐ other applications
Submit | Ignore

Consent
Alice wants to know your location
[✓ Allow] Deny
Conditions
valid until end of today
max rate: once
max precision: 500m
☐ only with symmetric consent
Also allow for
☐ other people
☑ requests for regular updates
☐ other information
Submit | Ignore

Figure 7 - generalization of consent responses

Instead of allowing or disallowing one person access to one particular piece of context information, the user can also choose to allow or disallow (see Figure 7):

- more people to have access to this information
- the information to be shared multiple times
- access to more information than requested
- access to this information through other applications.

This can greatly reduce the number of consent requests.

The issue remains what generalizations to propose to the user. As the study from Patil et al. shows [14], default settings are okay in 80% of the cases. There is no reason to assume that this would not apply for the generalizations people want. We therefore expect that in most cases users will accept a default generalization that is proposed by the consent server.

Polite blocking

Since denying access to context may be considered 'rude', we adapt the 'polite blocking' concept. This means that a requester of context will simply be told the context is not available, independent of whether this is the case because there is indeed no context source that can provide this information, or if the context owner has not given their consent. In case of subscription based interactions between application and context infrastructure, the context requester will simply never get an update of the context.

Expiry of the privacy policies

Several studies have indicated that it is not only hard for users to pre-configure privacy policies correctly (e.g. [4]), but the validity of privacy policies also decreases over time. Palen et. al [13] write that "privacy management is not about setting rules and enforcing them; rather, it is the continual management of boundaries between different spheres of action and degrees of disclosure within those spheres" [5]. In other words, the relationships with other people and the extend to which people would like to share personal information in particular situations changes over time. We therefore introduced an expiry time for each provided privacy policy, which the user can specify with each consent response that is valid for more than once. When a privacy policy has expired the user is automatically triggered again when the associated context information is requested again.

Request-respond vs subscribe-notify

When an application requests context information from the context infrastructure it can do this via a request-response or via a subscribe-notify interface. In the latter case, the application will be provided with notifications when the context changes (or in a fixed interval). For request-response the user is straightforward when to ask for consent: when the request is received. For subscribe-notify however there are several possibilities: when the subscription is received or for each individual notification.

Since there is no obvious best choice, we decided to design the interactive consent to do both. When a subscribe message is received, we send a consent request to the user to which the user can decide to either grant one time access or to allow the application to get notifications when the context information has changed, in addition to the generalization as described above. By providing an expiry time, the user is able to allow the application to be notified of context changes only for a limited time period. The user can also indicate the maximum rate at which applications will be notified about context changes.

To be able to notify the applications on context changes, the context management service needs to be notified by the context sources itself. It therefore needs to register with or actively poll the context sources. Privacy policies ensure that privacy sensitive information is not disclosed, effectively discarding information produced by the context sources and wasting communication and processing resources. The collection of context information by the context management service from the context sources needs to be carefully thought through to find an optimal solution. A trade off needs to be made between the flexibility of privacy policies and the impact this has on the collection of the required context information. Interactive consent does not provide more flexible privacy policies but enables more dynamic privacy policy management, which also adds to this complexity.

USE-CASE

To validate the concept of interactive consent, and to determine its limitations, we have used it in a context-aware application called C-Meeting. C-Meeting stands for Context-aware Meeting, and makes regular Outlook™ meetings context aware by informing the meeting participants if the other participants are already at the meeting location, and if not, if they are travelling towards the meeting location and will arrive on time. Figure 8 gives an overview of the C-Meeting application architecture, and a typical flow of interactions between the main components.

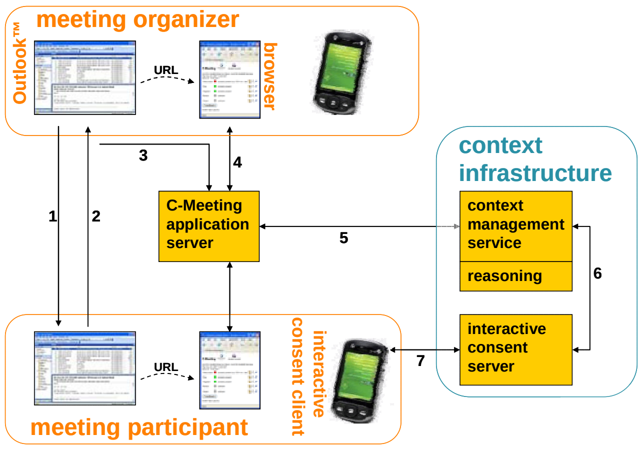
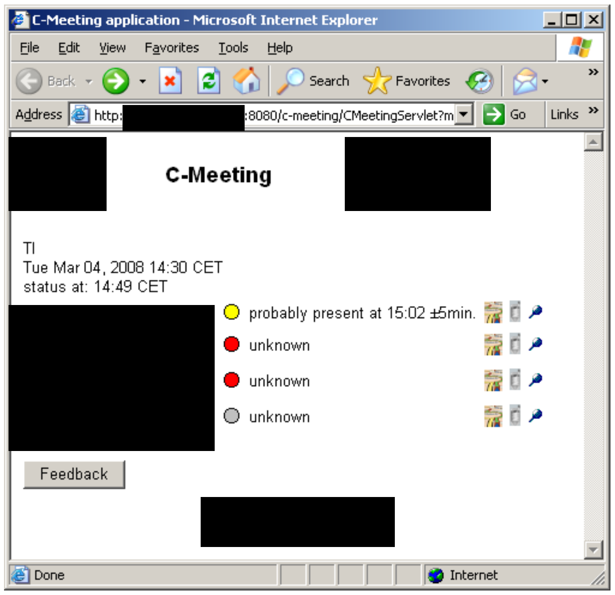

meeting organizer
Outlook™
URL
browser
context infrastructure
3
4
1
2
C-Meeting application server
5
context management service
reasoning
6
interactive consent server
7
URL
interactive consent client
meeting participant

Figure 8 - C-Meeting architecture

A *meeting organizer* can schedule a C-Meeting by scheduling a regular Outlook™ appointment with the category 'C-Meeting'. When the meeting request is sent (1) to a *meeting participant* a url is added to the request providing access to additional (context) information related to this meeting. Also a privacy statement is included that informs the participants that this is a context-aware meeting. The *C-Meeting application server* is notified of each participant that accepted the C-Meeting invite (2 & 3). Shortly before the meeting (15 minutes) any participant can view the likelihood of the other participants are at the meeting location (4). The C-Meeting application server asks the *context management service* for the relevant context information (5). Before sharing this information with the C-Meeting application service the context management services checks with the *interactive consent server* if the user has consented to this (6). The interactive consent server sends a consent request to the *interactive consent client* if the user has no stored privacy policy to cover this context request (7).


C-Meeting application - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http:[redacted]:8080/c-meeting/CMeetingServlet?m
Go Links
Feedback
Done
Internet

[redacted] C-Meeting [redacted]

TI
Tue Mar 04, 2008 14:30 CET
status at: 14:49 CET

[redacted]

- probably present at 15:02 ±5min.
- unknown
- unknown
- unknown

[redacted]

Figure 9 - C-Meeting user interface *(anonymized)*

Figure 9 shows the user interface of the C-Meeting application. For each of the participant (that have accepted the meeting request), the likelihood of that participant to be at the meeting location on time is indicated using 'traffic light' icons. By clicking on the small map icon on the right, participants can get a map showing the current location of the other participants.

The 'traffic light' icon is based on a prediction based on travel patterns (GSM cell ids), and has certain probability associated with it. Details on how this reasoning functionality works are out of scope for this paper. The probability is reflected in the text next to the icon. Based on the probability the likeliness of someone coming to the meeting location can be probably, maybe or probably not.

In this use-case, we can distinguish two different types of consent that we need:

1. When the meeting invite is received, and users have to consent if other participants can view their likelihood to be on the meeting location in time (including estimated-time-of-arrival)
2. When another meeting participant wants to view their actual location (only possible shortly before and during the meeting).

For the first type we found that our Interactive Consent mechanism was not the best solution. If one of the participants would open the C-Meeting user interface this would trigger consent request to all participants, which would be too intrusive. In addition, we expect that users will not consider the context information that is shared very privacy sensitive (i.e., if you are already at the meeting location, and if not, if you are travelling towards the meeting location and your estimated time of arrival). We therefore choose to make the consent here part of each meeting invite instead: this explicitly states the privacy consequences, and by accepting the meeting invite the user consents to this.

For the second type we did use our Interactive Consent mechanism. This is because the current location will be considered privacy sensitive information for most people. When one of the participants thus clicks on the map icon of another participant, this participant receives a interactive consent request (see previous section for screendump examples).

The C-Meeting application has been used over a period of 6 months by a small user group. A problem with the user experience of the interactive consent was that even if the user who's consent was requested would allow this, it takes too long from the perspective of the requesting user. Consistent with our polite blocking choice, the requesting user also does not know how to wait, since if we do not show a map this could be because the requesting user did not give their consent yet, or disallow access or because the location is unknown.

EVALUATION & CONCLUSION

Users should be offered personalized way to privacy control who can access their context, at what quality, and under what conditions. Interactive consent complements pre-configuration approaches to empower the user to have such control. Our interactive consent mechanism obtains explicit, informed consent from the user at the moment that context is requested. Benefits of this over pre-configuration approaches are improved understandability and fine-grained control because (i) the user is better aware which (context) information he or she consent to sharing, (ii) the user can better determine of usefulness of the application outweighs the lost of privacy given the context at that point of time and (iii) the consent can be provided at a per-context-request granularity.

The major disadvantage of our approach is the potentially unacceptable intrusiveness due the interruptions caused by the consent requests. To prevent this, we:

- Allow consent replies to be valid for longer periods, i.e., also for future context requests, thereby building up the privacy policies to enforce.
- Allow consent replies to be generalized, e.g., to apply to all colleagues and not only to the colleague who request some context.

Additional disadvantages are that the context requester may have to wait considerable time for the consent, making interactive consent less suitable for applications in which the context requester is a human that is waiting for an answer, or context applications that need a near immediate response such as emergency applications. In addition, as was the case of part of the consent needed in our use-case, there are application in which another form of consent is more natural for the user because it fits better in the normal interaction flow with the application.

Besides more elaborate user trials, future work will focus on reducing the intrusiveness further. A specific idea we have is to learn and propose generalized policies based on past consent replies, e.g., if a user always accept location requests from colleagues while working, to propose this as a policy. This may give better results than only proposing the same (default) generalizations to all users. A second improvement to reduce intrusiveness is to delay the consent request (when possible) to a time that a user considers it less intrusive, e.g., not during but right after a meeting.

ACKNOWLEDGMENTS

We acknowledge our colleagues for their contributions, especially Ingrid Mulder, Erwin Fielt,and Jacco Brok. This work is part of the Freeband AWARENESS project (http://awareness.freeband.nl), which is sponsored by the Dutch government under contract BSIK 03025.

REFERENCES

1. Barkhuus, L., Dey, A.K., “Location-Based Services for Mobile Telephony: a study of users' privacy concerns”, Interact 2003, Zurich, CH.
2. Beckwith, R., “Designing for Ubiquity: The Perception of Privacy”, IEEE Pervasive Computing, March, 2003.
3. Cornwell, J. et al., “User-controllable Security and Privacy for Pervasive Computing”, …
4. Cornwell, Jason, a.o., “User-Controllable Security and Privacy for Pervasive Computing”, Proceedings of the 8th IEEE Workshop on Mobile Computing Systems and Applications (HotMobile 2007), February 2007.
5. Gedik, B., Liu, L., “k-Anonymity: Architecture and Algorithms”, In IEEE Transactions on Mobile Computing, 7(1), January 2008.
6. Grudin, J., Horvitz, E., “Presenting choices in context: approaches to information sharing”, Microsoft Research, Ubicomp 2003 Privacy workshop.
7. Hsieh, G., Tang, K.P., Low, W.Y., Hong, J.I., “Field Deployment of IMBuddy: A Study of Privacy Control and Feedback Mechanisms for Contextual IM”, UbiComp 2007, LNCS 4717, pp. 91-108.
8. Janse, Maddy D., Vink, Peter, Soute, Iris, Boland, Heleen, “Perceived Privacy in Ambient Intelligent Environments”, In proceedings of Context Awareness and Trust (CAT 2007) workshop, IFIPTM conference, Moncton, NB, Canada.
9. Langheinrich, Marc, “Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems”, In proceedings of the Third International Conference on Ubiquitous Computing (UbiComp 2001). LNCS No. 2201, Springer-Verlag, pp. 273-291, Atlanta, USA, 2001.
10. Lederer S., Hong J.I., Dey, A.K., Landay, J.A., “Personal privacy through understanding and action: five pitfalls for designers”, In Proc. Personal Ubiquitous Computing (2004) 8: pp. 440–454.
11. Anonymized, “Enhancing privacy control through interactive consent: a user perspective”, submission to Privacy Enhancing Technologies symposium, Leuven, Belgium, 2008.
12. Nodder, C., “Say versus Do: building a trust framework through users’ actions, not their words.”, Ubicomp 2003.
13. Palen, L., Dourish, P., “Unpacking ‘Privacy’ for a Networked World”, CHI 2003, pp. 129-136.
14. Patil, S., Lai, J., “Who Gets to Know What When: Configuring Privacy Permissions in an Awareness Application”, CHI 2005, pp. 101-110.
15. Patrick, Andrew S., Kenny, Steve, “From Privacy Legislation to Interface Design: Implementing Information Privacy in Human-Computer Interactions”, In proceedings of PET 2003, LNCS 2760, pp. 107-124, 2003.
16. Pearce, Michael, Janssen, Craig and Narasimhan, Nitya, “Don’t Tick Off the User”, Position Paper, CMPPC Workshop, Pervasive 2007.
17. Petterson, John Sören et al., “Making PRIME Usable”, In Proceedings of the Symposium of Usable Privacy and Security (SOUPS), 4-6 June 2005, Carnegie Mellon University, Pittsburgh, PA, USA. July 2005.
18. Sinderen, M.J. van Halteren, A.T. van, Wegdam, M., Meeuwissen, H.B., Eertink, E.H., “Supporting Context-aware Mobile Applications: an Infrastructure Approach”, IEEE Communications Magazine, 44 (9). Sept 2006, pp. 96-104.
19. Sheikh, K., Wegdam, M., Van Sinderen, M., “Quality-of-Context and its use for Protecting Privacy in Context Aware Systems”, Journal of Software, Academy Publisher, 3(3), March 2008, pp. 83-93.
20. Weiser, Mark, "The Computer for the Twenty-First Century," Scientific American, pp. 94-10, September 1991.