Preserving Context with Context Capsules

Mike Spence and Siobhán Clarke

Distributed Systems Group
Department of Computer Science
Trinity College Dublin
Dublin, 2 Ireland
{spencem, sclarke}@cs.tcd.ie

Abstract

Mobile context-aware applications require sources of context information to drive more appropriate behavior. One valuable source of context information is context that already exists. Unfortunately, the situation an application is currently in and the situation in which a piece of context is initially used may differ by properties such as time and location. The further an application is away from the situation in which a piece of information was initially used, the more additional details it needs to be able to reuse it. In this paper we present the concept of a context capsule that facilitates the inclusion of the information necessary for context reuse.

Introduction

Advances in processor and battery technology have allowed non-trivial applications to move from stationary desktop environments onto mobile devices such as smart phones and PDAs. A consequence of the mobility afforded by these devices is that an application can no longer always assume the operating environment remains the same; therefore, adaptive mobile applications should acquire information about the surrounding environment in order to adjust their behavior appropriately.

Applications that use information from the environment to drive more appropriate behavior are called *context-aware*. *Context* is any information that can be used to characterize the situation of an entity (Dey and Abowd 2001). Context-aware applications acquire this information from attached sensors, other devices in the environment via wireless connections, and the user. The more context information an application obtains, the clearer the picture of the surrounding environment becomes. This makes it easier for an application to enact appropriate behavior for the current situation.

Unfortunately, the acquisition and processing of context information can be expensive or sometimes impossible, especially for a limited-resource device. It is normally infeasible to attach a sensor to a device for every relevant type of context and some calculations require too many resources to be carried out efficiently or at all. Necessary information sources can also become unavailable, e.g., a device can travel out of communication range with a wireless temperature sensor or a local GPS sensor can shut off due to a lack of power. In these cases, the necessary context information must be reused from other sources. Processed context information that is acquired from other user devices, termed *collaborative context*, is one such source. Another source is previously used context from the local device's context storage, termed *context history*. *Context reuse* then is the use of a piece of context outside the situation in which it was initially used. The reuse of already existing context offers an important additional source of context information; therefore, it aids an application's ultimate goal of providing more appropriate behavior.

In the remainder of this paper, we define context reuse and introduce *context capsules* that facilitate it. A context capsule combines an artifact, which can be a piece of context or any other application-exploitable object, with the information necessary for its reuse. Context capsules borrow from the notion of a time capsule. Much like a time capsule uses its contents to paint a clearer picture of the time period during which its contents were used, each of the contents of a context capsule, other than the artifact itself, help paint a clearer picture of the context in which an artifact was initially used. After introducing context capsules, we discuss the challenges related to implementing them and describe our current work to address these challenges. Finally, we discuss related work.

Context Reuse and Context Capsules

In the following subsections we define context reuse in more detail and present the structure and usage of context capsules that facilitate the reuse of context and other application-exploitable objects.

Context Reuse

It is not always possible for an application to generate a type of context that is necessary for the application to function. Additionally, generation may not be time- or resource-efficient as a piece of context of the desired type may already exist on another device or in the local context

history. In these cases, it is more advantageous to reuse that context information. Context reuse is defined as the use of a piece of context outside the situation in which it was initially used[1]. We characterize an "outside" situation as one that is either:

1. on the local device that initially used the piece of context but at a later time than the first use, or
2. on a different device than the one that initially used the piece of context

Use of a location reading generated ten minutes ago or an "In a meeting" activity that was applied the day before are examples of reuse for the first type of situation. Alternatively, acquisition of collaborative context, such as a location or an activity from another device, is an example of the second type of situation.

Because reuse requires determining the applicability of the piece of context from an "outside" situation to the current situation, the major impediments to reuse are interpretation complications. *Interpretation* describes the meaning an application assigns to a piece of context, which in turn affects its use. For example, an application might assume that a given location represents its current location in the absence of any other supplemental context (though the location might actually be from a situation outside of the current one, e.g., the user's location from ten minutes ago or the location of another user). In order for a piece of context to be disambiguated and to be correctly reused, an application must know the applicable situation for that piece of context. The concept of context capsules in the next subsection is an attempt to ground a piece of context and lessen the ambiguity that might prevent it from being reused properly.

Context Capsule Structure

The Oxford English Dictionary defines a time capsule as "a container used to store for posterity a selection of objects thought to be representative of life at a particular time." Each object in a time capsule gives a clearer picture of the moment at which the capsule was created. Like a time capsule, a *context capsule* stores an artifact, along with information that helps to put that artifact into context. Context capsules place all this necessary information in a self-contained representation that facilitates reuse.

The context capsule *artifact* is defined to be any application-exploitable object. Although so far we have only presented pieces of context as artifacts, some other examples include application behavior, a note, an audio file, or a document. These other types of artifacts face the same interpretation complications as a piece of context. For example, a document might only be appropriate at a specific location or a behavior might only be appropriate for a certain user at a certain time. Much like pieces of context used as artifacts, the reuse of non-context types of artifacts can also be facilitated by using context capsules. Any type of artifact can be placed in the capsule along with additional context information that describes the situation in which the artifact was used. These additional pieces of context are called S*urrounding Context Information (SCI)*. The SCI helps place the artifact in a context that allows it to be reused more effectively by both the device that initially used the artifact and other devices that acquire the artifact's corresponding context capsule.

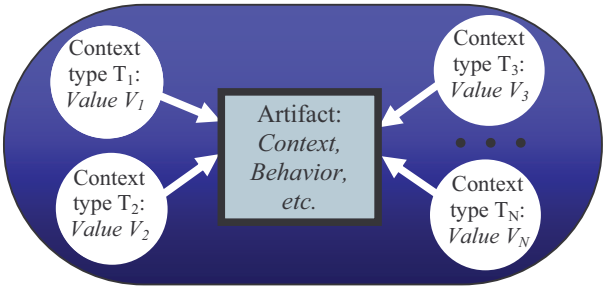

Figure 1 – Context capsule structure

In Figure 1 and the following context capsule figures, a capsule's artifact is designated by a rectangle, SCI by a circle, and attachment by an arrow from the SCI to the artifact. The value of a piece of context information can either be a single value, a range of values, or multiple values of both kinds. For example, a value for a time context can be 10:30 pm or all times from 12:00 pm to 1:00 pm. Alternatively, multiple values of both kinds can be used, e.g., a time context with values 10:30 pm, 10:45 pm, and all times from 12:00 pm to 1:00 pm.

Context Similarity and Context Capsule Usage

Context similarity is an important aspect of context capsule usage. Rarely are situations exactly the same, even if they are similar. For example, Figure 2 depicts the situation of a user at position A at 11:55 am on Wednesday (Situation 1) and the user two meters to the right at position B at noon on the next day (Situation 2). Even though Situation 1 is not exactly the same as Situation 2, they can be called similar and behaviors appropriate for one situation can probably be applicable to the other. In order for a capsule's artifact to be activated in a certain situation, there must be a notion of similarity to the situation in which it was initially used. We term the result of this comparison *context similarity*. Determining context similarity can be as simple as comparing two single pieces of context. For example, a location might be similar to another if they are within three meters of each other or if they are in the same room. If the SCI's value is a range, contexts can be judged similar if a current context's value falls within that range. Determining context similarity can also be as complicated as comparing a subset of the pieces of context defining a situation. In Figure 2, the situations might be similar because the user is the same, the positions are just two

[1] A piece of context can also be reused if it is generated and not used immediately. To simplify the discussion, however, we will include the piece of context in this scenario under context information that was *initially used.*

meters apart, and the times of day are within five minutes of each other.

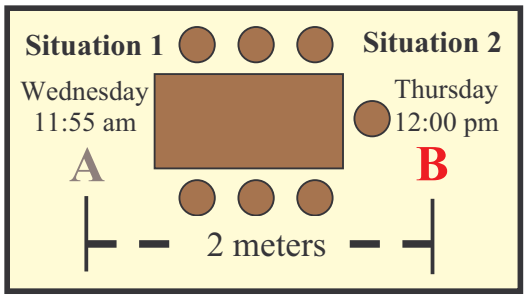

Figure 2 – Meeting room example

With this notion of context similarity, each SCI from a context capsule can be compared to the current context to see if it is similar. If this is the case, the artifact is seen to be applicable and the artifact is said to be *activated*. Alternatively, if a surrounding piece of context of an activated artifact is no longer similar to the currently sensed context, that artifact can be *deactivated*. Activation and deactivation allow an application to determine when and for how long an artifact is valid or invalid.

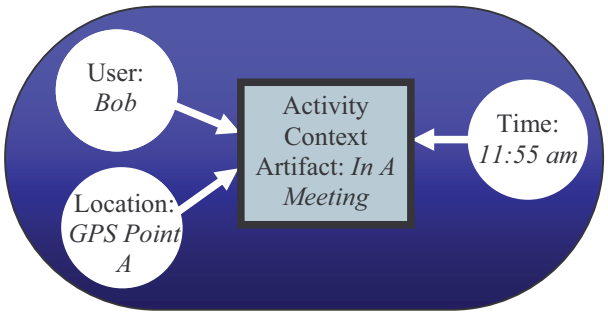

Figure 3 – Instance of a context capsule for Situation 1

As an example of context capsule usage, consider the user in Figure 2. While in Situation 1, he tells his application that he is in a meeting. The application then notes the context of that situation and creates the capsule in Figure 3 with an "In a meeting" activity context artifact. The next day, the application is presented with Situation 2 where the user is at location B at noon. Because the current context is similar to that of the capsule in Figure 3, the "In a meeting" artifact is activated and behavior appropriate for a meeting is applicable. This behavior might be subsequently activated by another capsule with that behavior as the artifact and the activity "In a meeting" as its SCI. Once the situation ceases to be similar, the "In a meeting" artifact is deactivated. This can occur if the user travels outside of the similar location range of the capsule or the current time is no longer similar to the capsule's time value. This activation and deactivation are done in Situation 2 without the user having to explicitly tell the application the activity was "In a meeting."

Context Capsule Benefits

The advantages of the context capsule representation and activation scheme are its aids to automatic manipulation, user manipulation and explanation, and artifact-SCI encapsulation.

Automatic manipulation. In the preceding example, the user did not have to explicitly create the context capsule. The user just told the application the current activity, and the application automatically created the capsule with that activity as the artifact and the current context information as the SCI. Similarly, any time an interesting behavior is invoked or document retrieved the application can create a context capsule with it as the artifact and the current context information as its SCI. The application might also modify capsules based on feedback from the user or inductive learning based on several capsules of the same artifact type. The possibility for the automatic creation and manipulation of context capsules necessitates less distraction and fewer tedious tasks for the user.

User manipulation and explanation. The user might also wish to define which situations are similar for an artifact by explicitly creating a capsule and specifying the SCI. This could be desirable if an application has not yet activated the artifact in the desired setting or a user desires specific values to be set in the SCI. Similarly, a user may wish to modify the components of a previously defined capsule, such as the types of SCI considered, the values of the SCI, or the artifact type and value. The causal nature of context capsules makes them easily understandable by humans and facilitates these user manipulations. For example, if a user wants to understand why an artifact was just activated, he needs only examine the SCI types and their values and compare them to the current situation. This representation also offers the user a method to easily modify the capsule if this artifact activation is undesirable.

Artifact-SCI encapsulation. Context capsules encapsulate the artifact and all the information necessary to reuse that artifact. Capsules can be stored on the local device and an application need only examine the contents of the capsule to determine its applicability. This property also allows the capsules to be passed to other devices with minimal preparation, as all the information needed to reuse the artifact is already contained in the capsule. The encapsulation also allows context-aware behavior to be passed to other devices. Capsules with behavior artifacts contain explicit mappings of context to behavior, as the SCI is the driving context information for the behavior artifact.

As presented, context capsules can facilitate the goal of generic artifact reuse; however, as the following section demonstrates, there are many challenges to reaching this objective.

Context Capsule Challenges

There are a number of challenges related to aiding context and other artifact reuse with context capsules on mobile

devices. These challenges can be separated into the following categories:

Selecting the Surrounding Context Information. Without an explicit mapping between the artifact and the context types that drive each type of artifact, it is difficult for an application to choose the appropriate subset of SCI types in a general manner. There is the possibility of including context types that have no bearing on the artifact and not attaching context types that do. This could lead to an artifact being activated or not activated erroneously. As an added difficulty, sometimes the context type to artifact type mapping isn't always fixed. For instance, although most activities have a fixed location, the activity of making a call from a mobile phone might not require a location. Furthermore, non-context types of artifact such as documents or audio files rarely have the same SCI types for each different instance of that artifact type.

Storage considerations. Storage is an important issue as mobile devices normally possess neither the memory required to store an extensive context history nor the resources necessary to efficiently reason over it. As with all historical context, it is not straightforward to store the SCI for each capsule efficiently. A compromise must be reached between efficient access of a time-indexed context history and efficient space utilization of storing only the SCI that is needed.

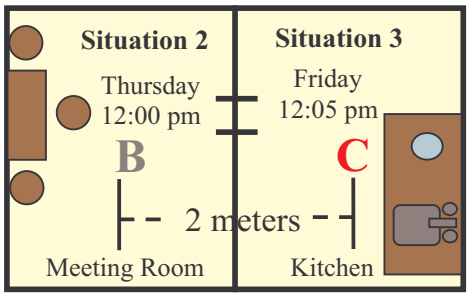

Figure 4 – Context similarity challenge

Defining context similarity. Generally determining similarity is difficult, as it may be measured differently across different situations. For example, Figure 4 shows Situation 2 from Figure 2 and a new situation in the office kitchen five minutes later on the following day (Situation 3). Situation 2 shows a user in a meeting, while Situation 3 shows that same user having lunch. While this is the identical situation transformation (in terms of magnitude) as in Figure 2, it is incorrect to assume that the user in Situation 3 is in a meeting. Alternatively, higher-level context may be used, such as a symbolic location instead of GPS location. In Figure 4, the capsules would cease to be similar because the Meeting Room is a different symbolic location than the Kitchen. Unfortunately, it is difficult for an application to determine the appropriate granularity of the SCI values and types.

Heterogeneous application environments. It can be difficult to use and reason over the context capsule, even if the SCI has all the information required for reuse. Different devices may support different reasoning methods, context types, and behaviors. Robustly structuring a context capsule so an application can still use the capsule even if it only understands a portion of the elements can help alleviate this difficulty. Additionally, context similarity that is not uniform across all applications may cause an application to judge incorrectly that a previously valid capsule is appropriate.

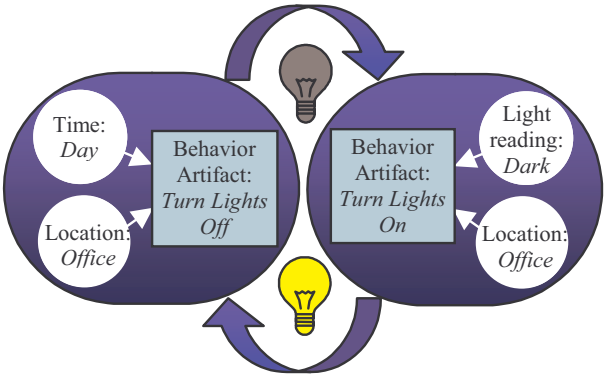

Figure 5 – Undesired emergent behaviour from two context capsules

Emergent behavior. Both local and remote devices can introduce context-driven behavior to the local device by means of context capsules with behavior artifacts. Unfortunately, even systems with the simplest of rules can exhibit seemingly random behavior from their interaction, as demonstrated (Wolfram 2002). Capsule interaction can cause unwanted application behavior, even if the capsules by themselves are appropriate for the user. For example, the two capsules with behavior artifacts in Figure 5 might cause the lights to flicker on and off on a dark, cloudy day. This undesired behavior emerges even though each capsule separately is consistent with the user's preferences. Emergent behavior like this is difficult to predict and prevent without severely limiting the introduction of capsules onto the device. Furthermore, this only becomes more complicated as the number of capsules increase.

Privacy and trust. Context capsules are formed by attaching context information to an artifact. While this additional context information may be necessary for artifact reuse, the context capsule also exposes that extra information to others if it is sent to another device. Privacy constraints must be applied to both the artifact and the SCI. As with other privacy policies governing context distribution, the most useful contexts may not be available under the current privacy policy. In this case, mappings of context types to artifact type and use of those mappings must be flexible. Sharing context capsules also adds an additional layer of complexity to the trust model. This complexity is beyond that of simply sharing context, as capsules can also be behavior mappings. An application needs to have a high degree of trust in the source and the capsule as even one malicious capsule can make the application unusable.

Current Status

Our current work to aid context reuse is placed within the broader scope of defining a framework to support mobile context-aware application development in general. Our *Hermes* framework uses the notion of a trail which encompasses several types of application (Clarke and Driver 2004). A *trail* is an ordered collection of activities which have associated contextual information, such as location and valid activity times. A trail helps an application guide users through these activities based on their current context.

The *Hermes* framework presents developers with a useful tool for creating mobile context-aware applications. To facilitate context reuse, the *Hermes* framework contains a *communication* component to acquire and share collaborative context, a *context history* component to store previously generated values, and a *capsule manager* component to manage the context capsules. The capsule manager handles the creation of local capsules, the filtering of irrelevant collaborative context capsules, and the activation and deactivation of capsules in similar contexts.

Currently, the SCI types for each artifact type are chosen from predefined mappings of context types to artifact type. The similarity ranges are also fixed, e.g., a location is similar if it is within a pre-defined distance of another location. The behavior types are also limited in order to inhibit undesired emergent behavior. Work to generalize the above in order to handle new types of artifact and relax some of these constraints is ongoing.

Related Work

The idea of attaching context to artifacts is not new. Most operating systems associate a date, size, and creating user to every file. In the realm of context-awareness, contexts such as creation time, creating user, and location have long been attached to artifacts. The Stick-e Note architecture (Pascoe 1997) stores and presents text-based notes based on the current context. While it also mentions artifacts beyond just text, it does not discuss an implementation or the challenges associated with this generalization. Context capsules also go a step beyond this architecture and facilitate context sharing and collaborative context reuse.

Context capsule usage is very similar to case-based reasoning (CBR), which is defined as "…[solving] a new problem by remembering a previous similar situation and reusing information and knowledge of that situation" (Aamodt and Plaza 1994). We plan on leveraging current CBR methods to handle some of the challenges listed above. For example, a capsule implementation might use an XML representation as in CBML (Hayes, Cunningham and Doyle 1998) to cope with some of the issues in the "Heterogeneous application environments" challenge. The context capsule approach is, however, a minimalist variation (exemplar-based) due to the limited resources and disconnected operation of mobile devices. This means that there can be only limited knowledge-based reasoning and the case memory that is always available is restricted. The simplicity of our approach allows mobile devices to reuse experiences without the extra functionality that would disable them. Our approach also focuses on the context-aware domain and relies on storing more information in the case to facilitate reuse when shared with other devices.

Summary

The reuse of already existing context offers mobile context-aware applications an important additional source of context information. Context capsules can be used to facilitate this reuse, but there are several challenges facing any implementation. Selecting the appropriate SCI types and values, defining situation-appropriate context similarity, and preventing undesirable emergent behavior are just some of these. If these challenges can be met, context capsules have the potential to alleviate several of the interpretation complications facing the reuse of any artifact in an automated manner or at the very least in a way that can be clearly understood and easily manipulated by the user.

Acknowledgements. This work is funded by Intel Corporation. We are also grateful to Cormac Driver, Eamonn Linehan, Shiu Lun Tsang, and the anonymous reviewers for their comments on early drafts of this paper.

References

Aamodt, A. and Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7(1):39-59.

Clarke, S. and Driver, C. 2004. Context-Aware Trails. *IEEE Computer* 37(8):97-99.

Dey, A.K., and Abowd, G.D. 2001. Understanding and Using Context. *Personal and Ubiquitous Computing* 5(1):4-7.

Hayes C., Cunningham P., Doyle M. 1998. Distributed CBR using XML. In Proceedings of the Workshop: Intelligent Systems and Electronic Commerce, Bremen, September 15-17.

Pascoe, J. 1997. The Stick-e Note Architecture: Extending the Interface beyond the User. In *Proceedings of the 2nd International Conference on Intelligent User Interfaces*, 261-264. New York, NY: ACM Press.

Wolfram, S. 2002. *A New Kind of Science*. Champaign, IL: Wolfram Media.