Granular Context in Collaborative Mobile Environments

Christoph Dorn, Daniel Schall, and Schahram Dustdar

VitaLab, Distributed Systems Group,
Institute of Information Systems,
Technical University of Vienna,
Vienna, Austria
{dorn, schall, dustdar}@infosys.tuwien.ac.at

Abstract. Our research targets collaborative environments with focus on mobility and teams. Teams comprise a number of people working on multiple projects and activities simultaneously. As mobile and wireless technology advances people are no longer bound to their offices. Team members are able to collaborate while on the move. Sharing context information thus becomes a vital part of collaborative environments. However, challenges such as heterogeneous devices, connectivity, and bandwidth arise due to the dynamic nature of distributed, mobile teams. We present a methodology for context modeling and employ a framework that reduces costs such as computing information and usage of network resources by transferring context at relevant levels of detail. At the same time, robustness of the system is improved by dealing with uncertain context information. Our framework is implemented on an OSGi container platform using Web services for communication means.

1 Introduction

A team is a group of people working collaboratively on tasks and activities. In our scenario we consider distributed teams working on activities within projects. Several team forms exist (such as Nimble, Virtual, and Nomadic or Mobile) [1], each featuring distinct characteristics such as shared vision, goals, and time span of existence. In real life, a person may be member of multiple, heterogenous teams at once. In fact, a person may work on more than one task or activity simultaneously. This means that users need to handle a number of activities, switching back and forth between activity context and gather all information relevant for a particular activity. In addition to these challenges, users need to keep track of changes and update their knowledge as information may become updated. Having updated and relevant pieces of information available is essential for decision making and has major impact on how a person assesses a problem or situation. In our research we consider team dynamics and mobility aspects. Nomadic or Mobile teams naturally have laptops or hand-held sized devices. Not only network bandwidth is a scarce resource in a mobile/wireless setting, but also computing power needed to process information, and thus battery consumption

R. Meersman, Z. Tari, P. Herrero et al. (Eds.): OTM Workshops 2006, LNCS 4278, pp. 1904–1913, 2006.

of the device. Therefore, one important goal of our architecture is to provide only information needed considering the user's current situation and context.

In addition, using context information to establish team awareness is one major challenge in mobile, distributed teams we intend to solve. The technical and organizational facets of collaboration cover the whole life cycle of context information: sensing, aggregating, storing, provisioning, and reasoning. As these problems are independent of the actual semantic content of context, this work focuses on the general mechanism to manage, process, and access context information.

The next section (2) will outline the problem in more detail. Thereafter, we present our fundamental concepts in Sec. 3 followed by our proof-of-concept implementation (Sec. 4). We then compare our approach to existing work (Sec. 5) and conclude our paper stating future research issues (Sec. 6).

2 Problem Statement

Taking one team member consuming context information, let us look at these challenges in detail. Suppose a team member is part of multiple teams. This person is updated on activities or status of fellow team members, however, his/her current activity should also be considered and provided information customized accordingly. At the same time his/her role such as leader, expert, adviser, observer or regular member influences scope of needed information. In other words, the amount and level of detail, relevant at a particular moment, depends on the context and thus changes continuously. Other related approaches to the issue of relevance such as choosing the best time or the most suitable communication channel are potential extensions for the future. Context plays an important part in all three cases.

Furthermore, team members expect information to be reliable to some degree. This means that decision making should be supported by evidences and data that do not lead to wrong assumptions. Thus, uncertainty has to be taking into account and a fallback mechanism provided, if confidence of data is too low.

Finally, usage of mobile devices limits the amount of information that can be processed and viewed. Transferring only relevant pieces of context information reduces costs such as network bandwidth, but also increases device responsiveness and battery lifetime. Thus, large amounts of context data, which are produced due to the nature of highly dynamic mobile environment, need to be filtered to reduce information being exchanged.

Throughout this paper, we will use the term `context consumer` for indicating the person, device, or software that receives context information, while the object the context information is about is labeled `context entity`. Before we present our concepts, a scenario will motivate our approach.

2.1 Scenario of Distributed Teams

Suppose a scenario where we have two distributed teams that wish to exchange context information. Figure 1 depicts teams at Location `A` and Location `B`.

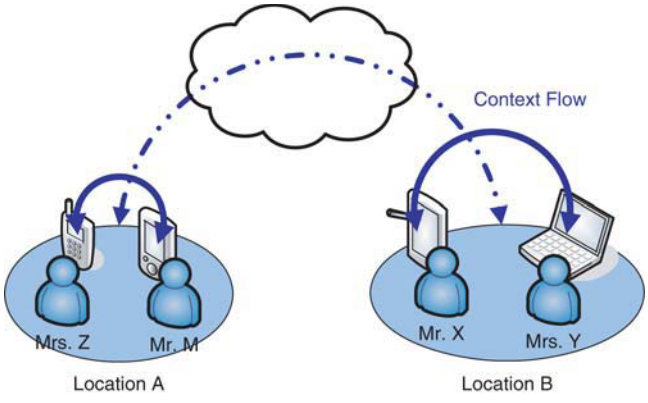

Fig. 1. Context Sharing in Distributed Teams

Team members may be in close proximity at a particular location but not collocated, for instance user `Z` and `M` at Location `A` at different floors in a building. Since both users work in the same team, it is required to share location information at a detailed level of spatial granularity. In this case both team members receive location information or updates at room level resolution, whereas the team located at Location `B` only needs to know course grained location information. Thus, higher level location information such as city and street is shared with remote teams. This not only saves bandwidth consumed by location updates, but also reduces unnecessary information being exchanged. The techniques to achieve this are presented in the next section.

3 Context Modeling and Processing

3.1 Hierarchical Context

The main idea of granular context lies in modeling a piece of information at multiple levels of detail. Consequently, the most generic information is found at the highest level, whereas the most detailed information resides at the bottom. Hierarchies consisting of levels are one way to model context granularity. One straightforward example is a location hierarchy, where continents or countries populate the upper parts and streets, floors, and rooms present the lower parts. Depending on the specific problem domain, such a hierarchy features additional levels at the top or bottom. Furthermore, for each level one or more values of varying complexity specify the context structure. To point out the difference between levels and values: the former one describes the granularity and position within the hierarchy, whereas the latter one contains the actual context information.

When it comes to handling actual context data, we pick up the outlined future work as presented in [2] and now directly include levels. Hence, there is no longer

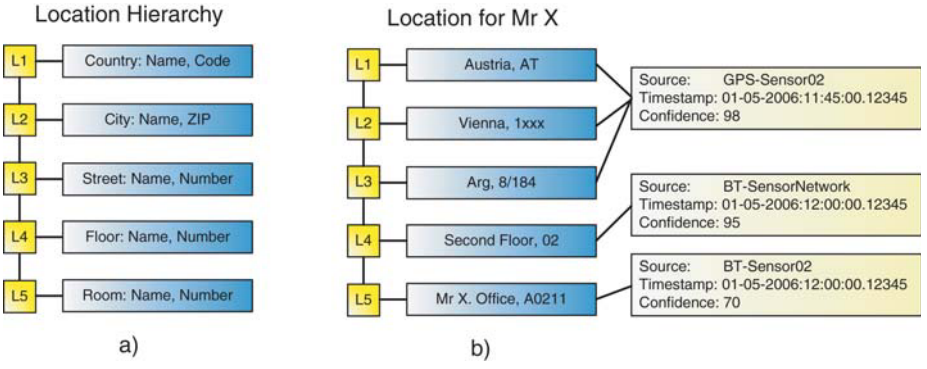

Fig. 2. Storing Hierarchical Location Information

a need for mapping hierarchies to other representational forms of context and vice-versa. Context in such a form allows navigating up and down along the hierarchical structure. In addition, context values in each level are tagged with the sensor source, timestamp of capture and data confidence. Fig. 2 visualizes our context model consisting of hierarchies and actual context information structured accordingly.

3.2 Empowering Collaboration

Structuring context as hierarchies provides the basis for our proposed mechanisms that tackle the outlined problems. Team members no longer receive the full amount of context information continuously, but just the information at the relevant level of detail. Determining what is relevant for a specific problem is outside the scope of this work, yet we present the means to manage relevance.

Furthermore, as outlined in the motivating scenario, varying context granularity is not only empowering awareness at the organizational level but also reduces the required bandwidth for context distribution at the link level. The amount of context flow between collocated team members is higher than between distributed ones. Thus, the sharing of context becomes completely relevance based as members a-priori define under which conditions data is exchanged.

Going beyond reducing bandwidth use, context granularity allows resource constraint devices to focus on their managable level of detail and thus limit processing and storing. Context hierarchies also provide a means to tackle unreliable context information. In order to utilize more coarse grained context information in case of too uncertain detailed levels, we require all confidence values within a hierarchy to be monotonically increasing. Consequently, instead of having to work with uncertain but detailed information, information that is less detailed but more confident allows for more appropriate actions. To establish a reliable confidence value for each level, we have to adapt the way context information is derived. Thus we will analyze the nature of hierarchies in the next subsection before focusing on processing.

3.3 Handling/Managing Hierarchies

When it comes to creating, filling, and managing, hierarchies differ to some extend from each other. First, we need to define the

- Scope for stating how many levels should cover what range of granularity
- Representational Form that decides if context is modeled in absolute values, relative values, numbers, or abstract concepts. Thus, context of a given granularity either consists of the whole hierarchy down to the respective level or each level contains all information.
- Sensors that provide raw context data. In the simplest case, each level is filled by a dedicated (logical) sensor. In contrast, only one sensor might fill all levels by providing the most detailed context. Finally, multiple sensors capture context for a single level.
- Inter-level Mapping Mechanism that describes how to derive the context for each level. Hence, where context information arrives at its most detailed form, values in all higher levels derive from this value.

Taking the location hierarchy as an example, the scope covers countries down to rooms. This also implies to use a representation form known from postal addresses (adding floors and rooms) split to five levels. As exemplary sensors, GPS and Bluetooth beacons provide the corresponding raw data. Finally, between every pair of adjacent levels, a mapping function defines how context information is transferred from one level to the other and vice-versa. For this example, the mapping function is simply adding the lower value or subtracting it. These four concerns are highly interrelated. If for example, purely GPS serves as a location technique, and location itself is merely provided in absolute coordinates. In addition, the levels reflect the precision of the data while the mapping function includes or excludes a digit of the coordinates.

Defining hierarchies that structure context of a single type such as location or time is rather straightforward. This process becomes more complex, once concepts from different domains are included that feature no natural ordering of granularity levels. For example, modeling team status including members, roles, activity and member distribution is nondeterministic, as it depends on the context consumer, whether information on collocated members or their activities describes more detailed information. In this case, either a predefined ordering of levels creates a static hierarchy, or context information (internal or external to the hierarchy) dynamically arranges the levels. In addition, advanced hierarchy forms consist of several other basic hierarchies, but are beyond the scope of this paper.

In contrast to conventional context systems, granular context potentially comprises multiple confidence values instead of a single one. Requiring all confidence values to grow monotonically from the most fine-grained up to the most coarse-grained level, each value reflects the accuracy of a piece of context information and not the sensor supplying raw data. Having confidence values at every level brings another advantage. The application using context information needs no

longer know about the implicit confidence characteristics of each sensor but can purely rely on the value for each level. Factors that influence the confidence values are manifold and depend heavily on the hierarchy's characteristics as outlined above.

- Sensor Model: Each sensor class has confidence values associated (e.g., statistically calculated) that describe the sensor's performance. Performance characteristics include a number of factors that limit the sensor's suitability for a particular application domain. Consider wireless sensor technology for context data acquisition. Results and the ability to obtain accurate estimates depend on the actual scene, e.g., spatial arrangement of furniture, dynamics such as open or closed doors, etc.
- Combining Heterogenous Sensors: In case of multiple sensors being used, sensors at different levels mutually improve the precision (e.g., GPS in combination with Bluetooth for localization).
- Context Entities: Accuracy of context data depends on a number of physical attributes and characteristics of the target entity we wish to observer (e.g., walking speed of a person, color, etc.).
- Computation: Accuracy and confidence of hierarchical context data depends on the inter-level mapping function (i.e., confidence increases if we step one level up).

Returning to the examplary location hierarchy, we are able to base our decisions e.g. on the context and confidence provided at the floor level, if the exact position of a person in a certain room (a more detailed level) cannot be determined accurately enough.

3.4 Storing Context

The challenge in storing granular context lies in preserving the hierarchy while enabling efficient access to the actual context value. Pure XML-based technologies provide one solution but are slow for larger amounts of data. We opted to use an object oriented database discussed in more detail in Section 4. Yet, as all our objects and hierarchy are described by means of an XML Schema we are able to compare and combine hierarchies. Allowing for multiple values in each level we can also extend a given hierarchy as described in our previous work [2]. Context itself is exchanged as XML documents also used by the sharing mechanims introduced in the following subsection.

3.5 Sharing Context

Polling for context changes is usually not a suitable option in a resource-restricted environment. Thus, sharing happens on an event basis. Interested context consumers subscribe for context changes in two ways. Using simple rules that are context content independent, they are notified about all changes in a given hierarchy, level or entity. Further restrictions can be made to take metadata such as

timestamp, source or confidence into account. Advanced rules are more complex as they cover either multiple entities, and/or are dependent on the context value of other hierarchies. They further allow notification if a certain context state is no longer valid. Unfortunately, we are unable to go into more detail here due to page restrictions. However, we will outline in more detail those rules in our future work where we intend to use RuleML combined with an interaction pattern-based approach.

To evaluate these concepts we designed and implemented a simple context framework targeted at a mobile collaboration environment presented in the following section.

4 Implementation

4.1 OSGi Based Architecture

Having discussed the basic building blocks of our approach we are now going to present our proof-of-concept implementation based on the OSGi container technology. We build a number of software services that are deployed as bundles. Hence they can be discovered and consumed by other bundles residing either in the same or remote containers. A loose coupling of software services using OSGi containers and Web services for communication means among containers allows us to implement a scalable and flexible infrastructure. Figure 3 shows a block diagram of the set of services and components that have been deployed in our infrastructure.

Context Information Acquisition. A Context Sensor Layer abstracts various context sources which can be either software sensors or actual hardware elements such as Bluetooth devices. Each sensor is wrapped as a logical entity, regardless of physical location, and controlled by the Sensor Layer. Measurements are fed into a Context Aggregation and Context Reasoning Component. A Context Reasoning Component is implemented for each context source. The Bluetooth context source requires a Bluetooth Context Reasoner as location measurements may be imperfect or ambiguous. Location information can be acquired from a number of heterogenous sources in order to increase confidence. In our implementation we make observations by scanning the environment with Bluetooth sensors (anchor nodes at well known positions). Mobile entities such as PocketPCs and Smartphones, also equipped with Bluetooth, respond to those frequent scans (inquiry method). Thus, we are able to localize entities by resolving the mobile devices location through anchor points. These anchor points are associated with a logical location which is provided by the environment controller (e.g., particular anchor node is located in Floor `Y`, Room `X`). The environment controller essentially holds all persistent information needed for our context-aware infrastructure, such as anchor nodes, users, devices belonging to a particular user, etc. In respect to Bluetooth based tracking, locating mobile entities indoors to room level granularity is usually sufficient for collocated teams.

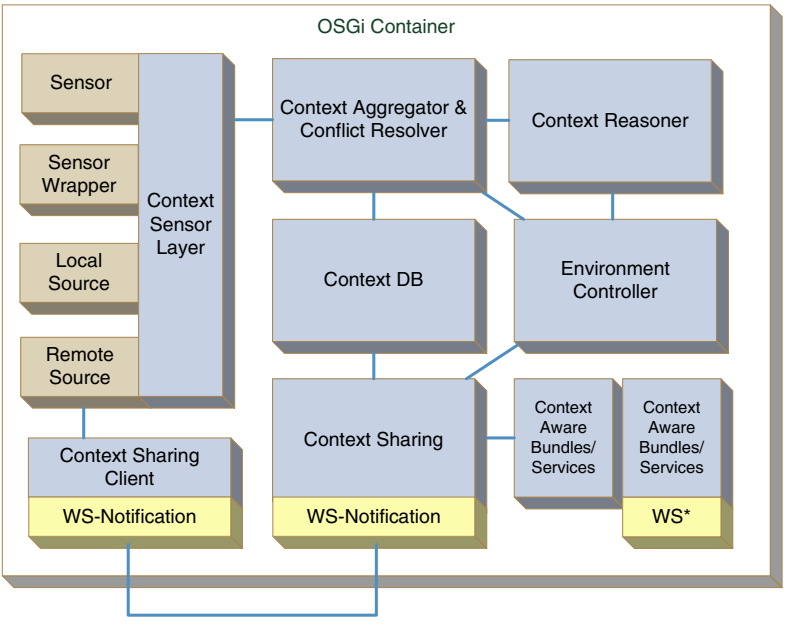

Fig. 3. OSGi Container Architecture

Context Reasoning and Conflict Resolution. The Context Aggregation Component fetches a hierarchy, depending on the context source, from the Context DB and updates the respective hierarchy level values. Each hierarchy holds context knowledge and confidence. In addition, time stamps allow us to save temporal versions of a specific hierarchy. By using previous versions of a hierarchy we can reason about plausibility of obtained information and thus smoothen state information and context. In our first prototype, we implemented three hierarchies relevant to collaborative environments. Besides the above mentioned location context, we are able to process hierarchies covering actions and reachability. The former one consists of levels describing `environment`, `subenvironment`, `project`, `artifact`, and `activity`, whereas the latter one consists of `online status`, `substatus`, `device status`, and `communication capability status`.

Context Database. An object oriented database as db4o[1] enables to dynamically change the context information to be stored; something that is much harder to achieve with a traditional relational database. In order to provide content independence the database requires all objects to implement interfaces for hierarchy, level and value. In doing so, the context reasoner extracts instances of specific context objects while the sharing component is able to focus on levels rather than context content.

[1] http://www.db4o.com

Subscription and Notification. As external access to context information happens through the WS-Notification interface, all data within a hierarchy and the hierarchy structure itself need to follow an agreed upon XML schema. We employed JAXB[2] (Java Architecture for XML Binding) to transfer Java objects into XML and vice versa. Hence, when a context change event occurs, the Context Sharing bundle checks for matching subscription at the occurring level or below. The modified part of the hierarchy structure is transferred into the XML format and then respective WS-Notification clients (at remote locations) are notified.

5 Related Work

Part of this paper builds upon our previous work [2] which was focused on a context sharing architecture for mobile Web services. In this paper, we improve the understanding of hierarchies and include a discussion on how to model and handle hierarchies. Furthermore, in this pervious work, the architecture consisted of an add-on to an existing context system, whereas here we take granularity a step further by having hierarchies as an underlying data model for all components involved in the lifecycle of context information. In particular, we introduce the notion of multi-level confidence and its calculation.

Of the many context-aware systems and frameworks, few support distributed context consumers. For a more detailed survey and overview refer to Baldauf et al. [3]. Besides centralized systems, distributed approaches such as the Solar middleware by Chen and Kotz [4] or the WASP Subscription Language by Costa et al. [5] target also mobile devices but lack the support for efficient sharing and processing. Other subscription enabled context frameworks include work by Sørensen et al. [6] and Hinze et al. [7]. Yet, Biegel and Cahill [8] present a framework for developing mobile, context-aware applications that uses a kind of context hierarchy. However, their concept has the notion of a task tree rather than structuring context information into various levels of detail.

A similar notion of context structuring is proposed by the ASC context model by Strang [9]. This ontology-based model introduces aspects, scales, and operations to semantically describe context information. Yet, the ASC model remains at a higher conceptual level, whereas our proposed concepts can be regarded as a light-weight, ready-to-use approach targeted at mobile collaborative environments.

Groupware system on the other hand, focus only on a subset of collaborative environments and issues and thus provide either very limited context support if at all, or target only narrow problem domains.

6 Future Work and Conclusion

We are at an initial stage of evaluating our prototype implementation. As presented in Section 4.1, three hierarchies for collaborative teams (describing a

[2] http://java.sun.com/xml/jaxb/

person's location, reachability, and activity) have been implemented and tested as proof of concept. Thorough evaluation and studies of our prototype as well as further improvements are subject of our future research. Furthermore, we will investigate techniques that build upon our framework for modeling and managing of relevance.

Yet, we believe that structuring context information in a granular way will greatly improve effectiveness of mobile, distributed teams as relevant data empowers team awareness, multi-level confidence ensure viable decisions and intelligent sharing reduces the strain on mobile devices.

Acknowledgment

Part of this work was supported by the Austrian Science Fund (FWF) under grant P18368-N04 Project OMNIS and EU STREP Project inContext (FP6-034718).

References

1. van Do, T., Jørstad, I., Dustdar, S.: Mobile Multimedia Collaborative Services. In: Handbook of Research on Mobile Multimedia. Idea Group Publishing (2006) 414–429
2. Dorn, C., Dustdar, S.: Sharing hierarchical context for mobile web services. Technical report, Vienna University of Technology (2006)
3. Baldauf, M., Dustdar, S., Rosenberg, F.: A Survey on Context-Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing (2006) forthcoming
4. Chen, G., Kotz, D.: Solar: An open platform for context-aware mobile applications. In: First International Conference on Pervasive Computing (Short Paper). (2002) 41–47
5. Costa, P.D., Pires, L.F., van Sinderen, M., Filho, J.P.: Towards a service platform for mobile context-aware applications. In: 1st International Workshop on Ubiquitous Computing - IWUC 2004. (2004) 48–61
6. Sørensen, C.F., Wu, M., Sivaharan, T., Blair, G.S., Okanda, P., Friday, A., Duran-Limon, H.: Context-aware middleware for applications in mobile ad hoc environments. In: ACM/IFIP/USENIX International Middleware conference 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing (online proceedings), Toronto, Canada (2004)
7. Hinze, A., Malik, R., Malik, P.: Towards a tip 3.0 service-oriented architecture: Interaction design. Technical report, Department of Computer Science, University of Waikato (2005)
8. Biegel, G., Cahill, V.: A framework for developing mobile, context-aware applications. In: Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. PerCom 2004. (2004) 361–365
9. Strang, T.: Service Interoperability in Ubiquitous Computing Environments. PhD thesis, L-M University Munich (2003)