Context-dependent Smart Space Formation in PECES[1]

Wolfgang Apolinarski, Marcus Handte, Pedro José Marrón

Networked Embedded Systems, Universität Duisburg-Essen
{wolfgang.apolinarski|marcus.handte|pjmarron}@uni-due.de

Abstract. Most existing middleware systems for pervasive computing enable the interaction between devices that are located within a limited spatial area such as a room or a building. Devices residing in this area can freely cooperate, for example, by sharing resources through service abstractions. In the past, this approach has been proven to be suitable for enabling smart meeting or class rooms. The PECES middleware extends this approach by building upon role assignment as a more expressive and uniform mechanism to form groups of interacting devices. Instead of fixing group formation to a particular spatial area, role assignment enables applications to form dynamic groups in a context-dependent manner. The groups can be composed hierarchically to allow the interaction of devices at distant locations connected via the Internet.

1. Introduction

Since the cost of providing communication middleware services such as device and service discovery or automatic application adaptation often depends on the number of interacting devices, existing middleware systems for pervasive computing exploit locality to improve their performance. To do this, they configure the execution environment by introducing logical boundaries that reduce the number of interacting devices. Usually, these boundaries are defined on the basis of proximity or location depending on the underlying system model.

Middleware systems that support **smart spaces** such as IROS [IROS], Gaia [GAIA], Easy Living [EASY], Aura [AURA] or Oxygen [OXYG] are usually bound to a specific geographic location. This location may represent a building such as a home or a work place [EASY], [OXYG] or a single room such as a meeting room or an office [GAIA], [IROS], [AURA]. Within this area, a coordinating server is responsible for providing additional services such as shared persistent storage, context management or application configuration, for example.

Middleware systems that support **smart peers** such as BASE [BASE] and PCOM [PCOM] or MundoCore [MUND], for instance, are usually relying on different proximity metrics. With these metrics they define the boundaries around each device, for example, as the set of devices in n-hop neighborhood [NEIG]. Similarly, concepts such as abstract regions [AREG] and scenes [SCEN] use location as reference point to form dynamic environments around it. For defining the boundaries, these approaches are limited to topological or geographical regions. Logical neighborhood [LOGI] and hood [HOOD] are two approaches that restrict the scope to the physical (i.e. 1-hop) neighborhood.

Although, location and proximity are often useful criteria to limit the number of interacting devices, the sole reliance on spatial characteristics can become an artificial barrier for applications that require cooperation with devices at distant locations. As mitigation, researchers have developed additional concepts such as Hyperspaces [HYPE] or Superspaces [SUPE]. However, such concepts complicate middleware and application development and they may result in too large spaces which may negatively impact the middleware and application performance.

In the Pervasive Computing in Embedded Systems (PECES) European research project, we have developed a more expressive, uniform middleware mechanism to support the context-dependent formation of smart spaces beyond the boundaries of a single spatial area. The core of this mechanism is formed by generic role assignment. Next, we outline the basic idea as well as its implementation.

2. Generic Role Assignment

The basis for generic role assignment is a set of devices that can communicate with each other. We assume that each device has some a priori knowledge about its context and that it is able to perceive

[1] This work has been partially supported by CONET (Cooperating Objects Network of Excellence) and PECES (Pervasive Computing in Embedded Systems), both funded by the European Commission under FP7 with contract numbers FP7-2007-2-224053 and FP7-224342-ICT-2007-2.

parts of its context at runtime. Given that a lot of context such as the device type and owner for example are usually static and that more dynamic context such as the device's location can often be acquired automatically by means of built-in sensors or by retrieving sensor values from other devices, this assumption can be fulfilled by most mobile devices today. Furthermore, we assume that the context of a device is stored locally, so that it can be accessed when needed.

Based on these assumptions, generic role assignment uses the device's context to assign roles. A **role** is essentially a tag that can be assigned to one or more devices. By definition, a role is assigned to any device as long as there are no further constraints that limit the assignment. To enable the automated computation of an assignment that reflects a desired group, we introduce rules. **Rules** define contextual constraints on the assignment of roles to devices.

In the PECES project, we have focused on three types of rules. A **filter rule** constraints the set of devices to a set of devices that exhibits a particular context. An example for such a filter rule is to demand that all devices should be at a certain location. A **reference rule** references other role assignments. Such a rule may demand that a device must exhibit a particular role, for example. As depicted on the left side of Figure 1, by using reference rules, it is possible to assign roles hierarchically. An **authentication rule** may express requirements to enforce the authenticity of a particular role assignment used within a reference rule or it may specify requirements on the authenticity of the context used within a filter rule. Thus, authentication rules can be used to ensure that role assignments are trustworthy and can be used for security critical purposes.

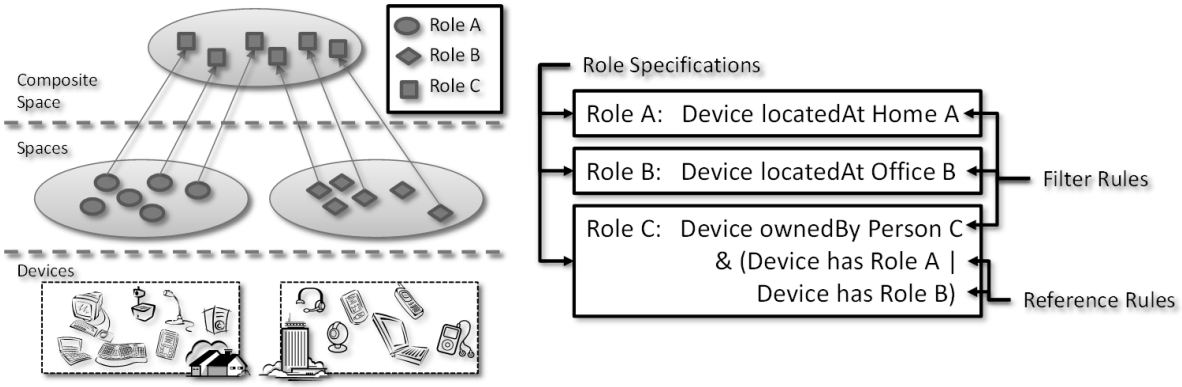

Figure 1: Hierarchical Role Assignment Example (left) and Role Specification Example (right)

The set of roles together with their corresponding rules form a role specification. To express a more complex assignment logic, a single role may be constrained using several rules that are combined using the logical AND and OR operators. The logical NOT operator is not supported as this avoids the evaluation of all globally connected devices which would not scale well.

Given that the necessary contextual information is known to each device, we can automatically assign roles to the devices whose context satisfies the constraints specified by their rules. Once the roles are assigned, they can be used directly by applications, e.g. to define the set of devices that shall perform a certain task, or they can be used in middleware services, e.g. to define the boundaries of hierarchically composed smart spaces.

A simple example on how role assignment can be used for hierarchical smart space definition is shown in Figure 1. The three role specifications depicted on the right define roles with associated rules. The specification that defines Role A consists of a filter rule that filters for all devices that are known to be at the location Home A. Similar to that, the specification with Role B consists of a filter rule for devices at the location Office B. In contrast to these simple specifications, the third specification is composed of different rules that are combined using logical operators. The first rule is again a filter rule that filters for devices belonging to Person C. This rule is linked with the AND operator with two reference rules which are, in turn, linked with the OR operator that reference Role A and B.

Intuitively, the runtime evaluation of the three role specifications will form three groups of devices, as depicted in Figure 1. The first group consists of devices that are located at Home A, the second group consists of devices located at Office B, and the third group conceptually consists of devices that are either located at Home A or Office B and belong to Person C. However, instead of specifying Home A and Office B explicitly, the role specification containing Role C refers to the specifications

with Role A and Role B. On the one hand, this enables the flexible composition of role specifications that may not be defined (completely) at design-time. On the other hand, it also enables a more efficient evaluation of role specifications at runtime, since the set of devices that can possibly receive Role C can be restricted to those devices that already contain Role A or Role B.

3. Implementation

Over the past two years, we have designed and implemented a generic role assignment system as core abstraction provided and used by the PECES middleware. The PECES middleware is written in Java and built on top of BASE [BASE] which enables spontaneous interaction between devices that are in the vicinity of each other. PECES extends BASE with additional communication support to allow the interaction of distant devices over the Internet and it realizes the set of middleware services depicted on the left side of Figure 2.

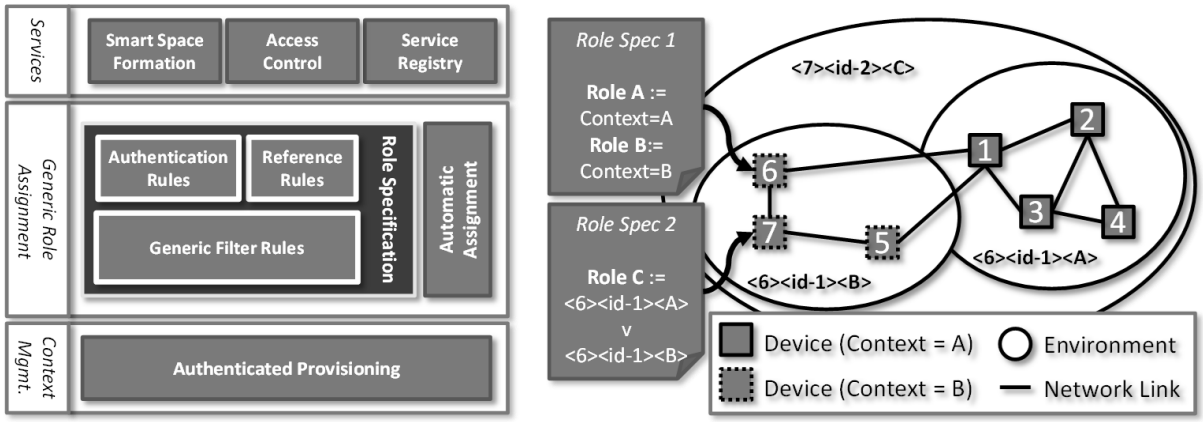

Figure 2: High-level Middleware Architecture (left) and Role Assignment Interaction (right)

At the lowest layer, each device is equipped with an OWL/RDF-based context storage that enables authenticated context provisioning. At the role assignment layer, a framework enables application developers and system administrators to define role specifications which use a subset of SPARQL for defining filter rules and our peer-based authentication framework [PEER] to define and verify authentication rules. To resolve reference rules, the middleware makes use of a distributed, hierarchically organized assignment registry. Based on this, an assignment algorithm is able to automatically execute role assignments which are then used by further middleware services.

To clarify this architecture, we describe the runtime interaction of its components in the following. To configure an environment or an application, a middleware service may start a role specification by sending it to a device equipped with an assignment service that can perform automatic role assignment. Since multiple role specifications may use the same role identifiers, the role assignment service first creates a globally unique id for the specification. This enables the unique identification of individual roles which is required to reference a particular role. To do this, the role assignment service concatenates the BASE device id with a locally unique id.

Once the id has been assigned, the assignment service analyses the specification to determine whether the role specification references some other role specification by means of reference rules. If the role specification does not contain reference rules, the assignment service creates a list of all SPARQL queries that represent the filter rules. Thereafter, it sends a single batch query to all locally connected devices. Once the list of responses is returned, the role assignment service evaluates the Boolean expression over the rules and computes the assignment.

If the role specification contains reference rules, the assignment service forwards the specification to the assignment service that is executing the referenced role specification. If a role contains multiple references, the specification is forwarded to each referenced assignment service. The assignment service that receives the specification will then execute it locally. Thereby, it considers only those reference rules that reference local assignments. The other rules are simply ignored. After the assignment has been computed at the referenced assignment service, a list of candidate assignments is returned to the original assignment component. There, the candidate assignments are transformed into

final assignments. To do this, the assignment component may have to intersect or unify the candidate sets in order to compute the result in cases where multiple references are concatenated using a conjunction or a disjunction.

Once the final role assignment has been determined, the role assignment service notifies each device that receives at least one role. Thereby, the service transmits all assigned roles. Applications may register local listeners to receive changes to these assignments – which provides the basis for all mechanisms built on top of the role assignment system.

An example for this process is depicted on the right side of Figure 2. The figure shows 7 devices that execute two role specifications. The first role specification defines two spaces using the roles A and B. Both roles solely rely on filter rules in order to define the sets of devices. In order to keep the figure simple, we refrain from using SPARQL syntax, instead we simply assume that role A requires context A and role B requires context B. Once the role specification is started at device 6, the device assigns a unique id, i.e. <6><id-1>. Thus, the roles can be identified by concatenating the role specification id with the role name, i.e. <6><id-1><A> or <6><id-1><B>. Since there are only filter rules, the assignment component queries the context of the connected devices and computes the assignment according to the rules. Finally, the assignment component notifies all devices that received a particular role.

The second role specification in the example refers to the first specification to define an environment using role C that consists of all devices that have role A or B. When the role specification is started at device 7, the unique id is generated and the role specification is analyzed. Since the role specification contains reference rules, the role specification is forwarded to the devices that are managing the referenced specification. In this example, this is done by device 6. To determine the managing device, the device 7 can simply use the BASE id that is embedded in the reference. Device 6 then computes the candidate set consisting of devices with role A and role B and returns it to device 7 which performs the final assignment. In this example, the candidate set and the final set are identical. However, if several specifications on multiple devices are referenced, it may be impossible to determine the set locally on the referenced devices. Once the set has been computed, device 7 notifies all relevant devices.

On top of the assignment service, additional services implemented as part of the PECES middleware encompass a service to form hierarchical smart spaces in which devices can share resources, a role-based access control service that enables developers to define and enforce access rights using role specifications as well as a service registry that can be scoped using role specifications. To use roles as basis for such services, it is necessary to associate a particular meaning with a particular name. This allows the services running on the device to perform specific actions when assignments change.

For example, for smart space formation, the PECES middleware associates a specific meaning with two roles. A so-called MEMBER role is used to identify devices that reside within a particular, possibly hierarchically defined smart space. A GATEWAY role is used to identify devices that can provide connectivity to other spaces through the internet. The MEMBER role is then used to define search scopes for service lookups and the GATEWAY role is used to connect to other spaces. An application may use these roles, for example, to (re-)execute queries for available services whenever a new device receives a MEMBER role.

Similarly, for role-based access control, a developer must associate an application-specific meaning with the roles that are used to control access. For example, to provide a fine-granular access to a service, a developer might define a GUEST, USER and ADMIN role with different access privileges. By securing role assignment with authentication roles the developer can then use the PECES middleware to test whether a caller may execute a particular operation by determining its roles.

4. Conclusion

Most existing middleware systems for pervasive computing enable the interaction between devices that are located within a particular spatial area such as a room or a building. By building upon role assignment as a uniform, expressive mechanism to form groups of interacting devices, the PECES middleware enables a hierarchically structured, context-dependent formation of spaces that can extend to multiple distant areas. This enables a more flexible definition for smart spaces and it simplifies the development of pervasive applications that require interaction beyond a single smart space.

Within the PECES project, we are currently working on supplemental development tools that simplify the usage of role specifications by providing visual editors as well as an emulation environment that can be used for testing. Once the tools are available, they will become part of an open source release of the PECES middleware that is targeted to take place in late 2011. At the same time, we are working on a number of applications to validate the suitability of role assignment as a basic abstraction to support applications that require interaction beyond a single smart space. More information about the PECES European research project and middleware can be found on the project website at http://www.ict-peces.eu/.

References

[AREG] M. Welsh and G. Mainland, "Programming sensor networks using abstract regions," in NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation, Berkeley, CA, USA, 2004, pp. 3–3.

[AURA] D. Garlan, D. P. Siewiorek, and P. Steenkiste, "Project aura: Toward distraction-free pervasive computing," IEEE Pervasive Computing, vol. 1, pp. 22–31, 2002.

[BASE] Marcus Handte, Stephan Wagner, Gregor Schiele, Christian Becker and Pedro José Marrón, "The BASE Plug-in Architecture - Composable Communication Support for Pervasive Systems", In 7th ACM International Conference on Pervasive Services, July 2010.

[EASY] B. B. Brian, B. Meyers, J. Krumm, A. Kern, and S. Shafer, "Easyliving: Technologies for intelligent environments." Springer-Verlag, 2000, pp. 12–29.

[GAIA] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, "Gaia: a middleware platform for active spaces," Mobile Comp. Com. Rev., vol. 6, no. 4, pp. 65–67, 2002.

[HOOD] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler, "Hood: a neighborhood abstraction for sensor networks," in MobiSys '04: 2nd international conference on Mobile systems, applications, and services. NY, 2004, pp. 99–110.

[HYPE] J. Ma, L. T. Yang, B. O. Apduhan, R. Huang, L. Barolli, M. Takizawa, and T. K. Shih, "A walkthrough from smart spaces to smart hyperspaces towards a smart world with ubiquitous intelligence," in ICPADS '05: 11th International Conference on Parallel and Distributed Systems. Washington, DC, USA: IEEE Computer Society, 2005, pp. 370–376.

[IROS] S. R. Ponnekanti, B. Johanson, E. Kiciman, and A. Fox, "Portability, extensibility and robustness in iros," in PERCOM '03: 1st IEEE International Conference on Pervasive Computing and Communications. Washington, DC, USA: IEEE Computer Society, 2003, p. 11.

[LOGI] L. Mottola and G. P. Picco, "Using logical neighborhoods to enable scoping in wireless sensor networks," in MDS '06: 3rd international Middleware doctoral symposium. NY, 2006, p. 6.

[NEIG] G.-C. Roman, C. Julien, and Q. Huang, "Network abstractions for context-aware mobile computing," in ICSE '02: 24th International Conference on Software Engineering, NY, USA, 2002, pp. 363–373.

[OXYG] L. Rudolph, "Project oxygen: Pervasive, human-centric computing – an initial experience," in CAiSE '01: 13th International Conference on Advanced Information Systems Engineering. London, UK: Springer-Verlag, 2001, pp. 1–12.

[PCOM] C. Becker, M. Handte, G. Schiele, and K. Rothermel, "Pcom – a component system for pervasive computing," in 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom'04). Washington, DC, USA: IEEE Computer Society, 2004, p. 67.

[PEER] W. Apolinarski, M. Handte, and P. J. Marr´on, "A secure context distribution framework for peer-based pervasive systems," in PerWare Workshop at the 8th Annual IEEE International Conference on Pervasive Computing and Communications, March 2010.

[SCEN] S. Kabadayi and C. Julien, "A local data abstraction and communication paradigm for pervasive computing," in 5th IEEE International Conference on Pervasive Computing and Communications (PerCom'07), 2007, pp. 57–68.

[SUPE] J. Al-muhtadi, S. Chetan, and R. Campbell, "Super spaces: A middleware for large-scale pervasive computing environments, perware 04," in IEEE International Workshop on Pervasive Computing and Communications, 2004, pp. 198–202.