Awareness of Entities, Activities and Contexts in Ambient Systems

Bent Bruun Kristensen(✉)

Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark
bbkristensen@mmmi.sdu.dk

Abstract. Ambient systems are modeled by entities, activities and contexts, where entities exist in contexts and engage in activities. A context supports a dynamic collection of entities by services and offers awareness information about the entities. Activities also exist in contexts and model ongoing collaborations between entities. Activities and local contexts also obtain awareness information from the context about the dynamic collection of entities. Similarly activities, local contexts and entities are offered awareness information about activities and local contexts.

Keywords: Awareness · Entity · Activity · Context · Ambient system

1 Introduction

We focus on ambient systems—a combination of the reality-virtuality continuum, ubiquitous computing and robotics [1]—that are more complex than traditional systems and evolve more spontaneously. We understand ambient systems in terms of concepts and phenomena where users interact with several computations simultaneously and time and space aspects are essential. An ambient system identifies users, collaborates intelligently with the user, and supports users in their ongoing activities. We have evolved from "users communicating with information systems" to "users participating in ambient systems". Ambient systems support the participants' intentions and work tasks. However, the participants have the initiative and they deliver and request information. Users need models to understand and use ambient systems—and developers need models to design and implement ambient systems.

Ambient system models include entities, activities and contexts. A context supports a dynamic collection of entities by services and offers awareness information about this collection. Activities in contexts model ongoing collaborations between the entities. The notion of entity, activity and context used in this paper are very similar to the notion of tangible object, association and habitat used for modeling and characterization of ambient systems [1]. These concepts are abstractions with informational and physical aspects and in addition they supply each other and may be combined in descriptions and executions of collaboration. Tangible objects are autonomous and cover users, things, etc. Associations cover group activities between tangible objects like collaborations, meetings, etc. And habitats cover universes in which tangible

P.C. Vinh et al. (Eds.): ICCASA 2013, LNICST 128, pp. 144–156, 2014.
DOI: 10.1007/978-3-319-05939-6_15,

objects and associations exist—like rooms, places, etc. [2]. The system is dynamic i.e. the instances of tangible object, association and habitat may appear and disappear. Tangible objects engage in activities and both tangible objects and associations enter and leave contexts.

Awareness means "has knowledge of existence" or in dynamic situations "becomes aware of existence". Contexts support awareness: When an element enters/leaves a context any other element in the context is notified about this—as well as the element entering/leaving is notified about any other element already in the context. Awareness is well known for traditional objects—and in this respect entities are seen as objects. However, awareness for contexts and activities is original: Context and activity are abstractions distinct to objects and entities (not only objects representing activities or contexts)—the characteristics of contexts and activities imply additional awareness support. The aim of the paper is to clarify the notion, potential, and examples of awareness support of activities and contexts (as a supplement to existing awareness of objects). The paper also includes experiments with prototypical systems and experimental implementations of awareness especially for activities and contexts.

2 Background

Habitats are conceived as some kind of locality, delimited by some boundary, comprising inhabitants and providing support to its inhabitants in the form of opportunities and services that allow its inhabitants to interact and achieve their various goals [3]. Physical habitats are close to our intuitive understanding in describing the localities where organisms and life-forms grow and live. However characterizing man-made physical environments also as habitats, rather than mere spaces, introduces the notion of thinking about how inhabitants and habitats influence each other and evolve over time. Informational habitats are those spaces that are created with and exist in information. From the definition of habitat an informational habitat is some kind of locality with inhabitants, who draw upon the support of the locality and both adapt accordingly.

Associations support associative modeling and programming through abstraction from collaborations [4]. The association abstraction integrates activity and role aspects, where the activity is between autonomous entities. The directive (sequencing rule) of an association is a central, partial description of interactions of the participating entities. An entity is autonomous, i.e. only the entity itself may execute its methods. An entity executes its contributions (e.g. a method invoked by the entity) to the activity in the context of the entity. An entity participating in various associations executes contributions from the various directives interleaved.

Related Work. Software that examines and reacts to an individual's changing context is described in [5]. Such software promotes and mediates people's interactions with devices, computers, and other people, and it helps navigate unfamiliar places. The paper defines context-aware computing, and describes four categories of context-aware applications: proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions.

When the user's situation, place or activity change the functionality of devices adapt to these changes [6]. A layered real-time architecture supports this kind of context-aware adaptation based on redundant collections of low-level sensors. A personal digital assistant and a mobile phone are used with a prototype to demonstrate situational awareness.

By means of wireless information services any social institution may structure activity in any place and thereby break down the traditional mapping between institutions and places [7]. This complicates the analysis of context for purposes of designing context-aware computing systems. Context has a physical, architectural aspect, but most aspects of context will also be defined in institutional terms. The paper includes two conceptual frameworks for the analysis of context in mobile and ubiquitous computing.

Common architecture principles of context-aware systems and a layered conceptual design framework are used to explain the different elements common to most context-aware architectures [8]. The resulting context-aware systems offer entirely new opportunities for application developers and for end users by gathering context data and adapting systems behavior accordingly.

A survey including a general analysis framework for context models and an up-to-date comparison of the most interesting, data-oriented approaches available in the literature in [9] is motivated by context-aware systems pervading everyday life. The survey provides a comprehensive evaluation framework, allowing application designers to compare context models with respect to a given target application.

A review of selected literature of context-aware pervasive computing in [10] supports the use of theory and practice to enable anywhere and anytime adaptive e-learning environments. The review particularly elaborates on context, adaptivity, context-aware systems, ontologies and software development issues.

3 Awareness

A conference information system is an example of an ambient system: Physical contexts include `seminar room`, `reception` and `coffee room`. Entities include `attendee` and `speaker`. Activities include `session`, `meeting`, `coffee break` and `conference dinner`. The `conference` is an activity in which `attendees` participate and take on roles at various events at various locations. `program committee`, `session chairs`, `speakers` are examples of informational contexts. A software agent, `desk agent`, is an example of an informational entity that offers reminders about time and place, history overview and GPS guidance. The physical entity `service robot` moves around and offers refreshments and physical guidance to `attendees`.

Figure 1 illustrates a system (static view only) with entities, contexts and activities schematically: Rectangles, stars and circles illustrate physical and informational contexts, activities and entities, respectively. Contexts are related in various ways and entities may be engaged in several activities.

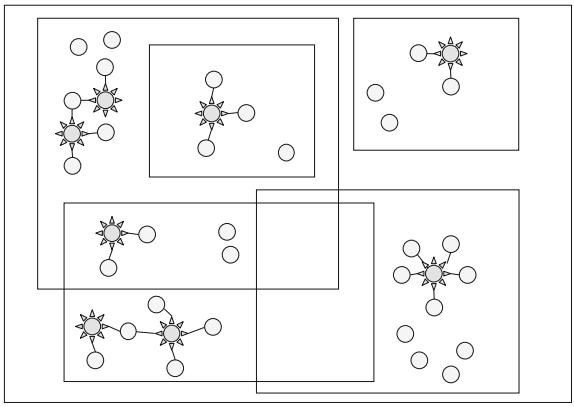

Fig. 1. Schematic example

Contexts: Services and Awareness. Contexts have two characteristic offers, namely service and awareness support:

- Services: A context has a number of services that are offered to entities contained in the context. The services are offered in various ways (in programming as methods local to the context to be invoked by the entities). For example `internet` and `printer` are offered to `attendees` in certain `working areas`.
- Awareness: Entities in the context obtain information about entities entering and leaving the context and vice versa. The information may be obtained in various ways and an entity may utilize this knowledge about other entities. For example an `attendee` that enters the `seminar room` for a `session` is offered information about the `attendees` already in the `seminar room` and vice versa.

We distinguish between descriptors and instances of entities, activities and contexts (similar to the distinction between classes and objects). From a description a number of instances may be created—and such instances have state. A context also contains (local) descriptors for entities, activities and contexts. In this way a context instance is not only a container of instances but by its local descriptors the context also offers potential additional instances to be created.

Figure 2 illustrates a context and entities `X`, `Y`, and `Z` schematically: Entity `Z` utilizes the services in the context, entity `X` enters the context and entity `Y` leaves the context. Entity `Z` becomes aware when entity `X` enters and entity `Y` leaves and entities `X` and `Y` become aware about the existence of `Z`. This awareness may be modified in

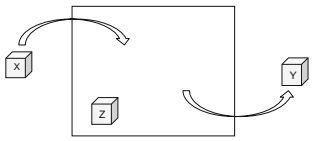

Fig. 2. Entity in, entering or leaving a context

various ways for example the notification is restricted to certain entities depending on type, state etc. Figure 2 also illustrates a generalized situation where the cube illustrates not only entities but also contexts and activities: An activity in the form of a `program committee meeting` may be interrupted shortly, moved into another location, and then continued. Similarly a context containing attendees with interest in `ambient systems` may be moved from context `attendees` to context `special interest`. Therefore awareness may include entities, activities and contexts in relation to each others, however for simplicity reasons this paper focuses specifically on awareness of activities and contexts in relation to entities.

Organization of Contexts. Contexts may be organized as follows:

- Disjoint: Two contexts are disjoint if no element is in both contexts: `seminar room` and `meeting room` are typically disjoint physical contexts.
- Overlapping: Two contexts are overlapping if some elements are in both contexts: `session chairs` and `speakers` are typically overlapping informational contexts.
- Nested: One context is nested within another context if any element in the context is also an element of the other context: `speakers` is nested within the `attendees` context.

Figure 3 illustrates the organizations of contexts schematically: Fig. 3(a) illustrates entities `X` and `Y` in disjoint contexts. Figure 3(b) illustrates entity `Z` in two overlapping contexts in which entity `X` respectively entity `Y` also exist. Figure 3(c) illustrates entity `Y` in a context that is nested in another context in which entity `X` also exists. Figure 3 also illustrates a generalized situation where the cube also illustrates not only entities but also context and activity.

The actual contents of context determine if the context is disjoint or overlapping: In Fig. 3(a) if an entity `Z` enters one of the contexts and then the other context then the organization changes from (a) to (b), and vice versa. For the nested organization in Fig. 3(c) all the contents of the local context is also in the enclosing context, i.e. entity `Y` is in both contexts whereas entity `X` is in the enclosing context only. The enclosing context offers services and awareness in Fig. 3(a). In Fig. 3(c) the local context typically has priority over the enclosing context. In Fig. 3(b) additional rules decide which context containing `Z` has priority.

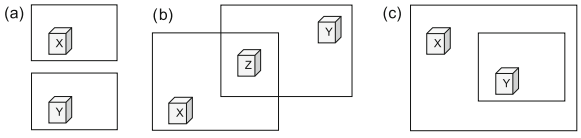

Fig. 3. (a) Disjoint, (b) Overlapping and (c) Nested contexts

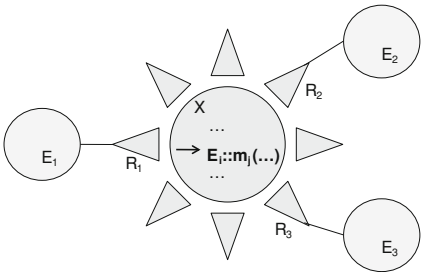

Fig. 4. Activity with directive and roles

4 Activities

Figure 4 illustrates activity X with its roles R_i, i=1…3, where entity E_i is engaged in X through role R_i. In the directive of X an arrow indicates the current position. The notation $E_i::m_j(...)$ is a request to autonomous entity E_i to execute its method m_j (because an entity is autonomous an activity cannot invoke its method but only request the entity eventually to do so) [4]. The state of X includes current position and knowledge of entities E_i, i=1…3.

Awareness of an activity includes notification of the state of the activity: An activity includes a directive, roles and entities engaged in the activity through these roles. The activity is dynamic, i.e. its state includes the current execution point of the directive. The state also includes the identity of (e.g. references to) the entities currently engaged in the activity. When an activity enters or leaves a context the state of the activity also enters/leaves. Awareness in relation to activities includes that the state of an activity is used appropriately in two situations:

- Activity awareness of entities.
- Entity awareness of activities.

Activity Awareness of Entities. An activity may be aware of entities: An activity enters or leaves a context including entities cf. Figure 5: The activity obtains information about entities in the context. Similarly an entity enters or leaves a context including activities cf. Figure 6: Also here the activities obtain information about the entity entering or leaving.

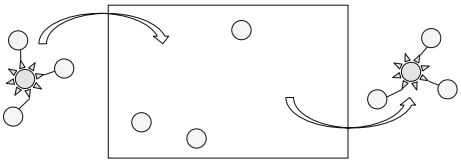

Fig. 5. Activity entering or leaving context with entities

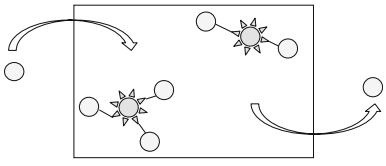

Fig. 6. Entity entering or leaving context with activities

At the conference various activities other than sessions take place: Imagine a session preparation activity of some session between speakers and the session chair. This activity may be initiated by email interaction in an informational context supporting this kind of work. When session chair and speakers are present at the conference the session preparation may for practical reasons be moved to a meeting room. Upon its entry to the meeting room the session preparation is informed about attendees already in the meeting room—including for example additional speakers at the session. The ongoing session preparation is informed whenever an attendee (maybe yet another speaker at the session) enters the meeting room.

Entity Awareness of Activities. An entity may be aware of activities: Fig. 6 illustrates an entity that enters or leaves a context containing activities: The entity obtains information about activities in the context. Figure 5 illustrates entities where an activity enters or leaves the enclosing context: Also here the entities obtain information about the activity entering or leaving.

An attendee entering a context seminar room with an ongoing session is informed about the session including e.g. session chair and speakers. The attendee immediately is informed about the actual state of the session including the paper currently being presented and the speaker. The progress is also available i.e. whether the speaker currently presents the paper or answers questions.

5 Contexts

Figure 7 illustrates context C_G containing entity E, activity A, and local context C_L. The state of C_G includes E, A and C_L. Contexts are similar to name structures [11]: The rules of *static scoping* applies except that the organization of the contexts is modified dynamically because entities, activities and context may dynamically may enter and exit contexts (distinct from *dynamic scoping* where the scope structure is determined by the calling sequence of e.g. functions).

Awareness of a context includes notification of the state of the context: A context contains a dynamic collection of entities, activities and local contexts. When a context enters or leaves a context the state of the/now local) context—including entities activities and local contexts—also enters/leaves. Awareness in relation to local contexts includes that the state of a context is used appropriately in two situations:

- Context awareness of entities.
- Entity awareness of contexts.

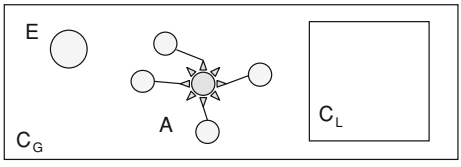

Fig. 7. Context with entity, activity and local context

Context Awareness of Entities. A context may be aware of entities: A context enters or leaves a context containing entities cf. Figure 8: This context obtains information about entities in the context. Similarly an entity may enter or leave a context containing contexts cf. Figure 9: Also here these contexts obtain information about the entity entering or leaving.

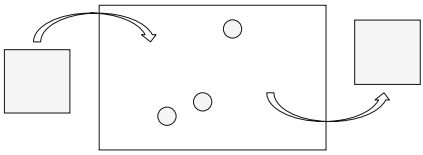

Fig. 8. Context entering or leaving context with entities

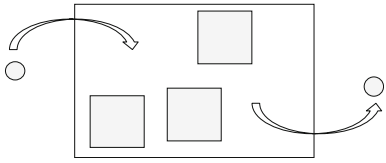

Fig. 9. Entity entering or leaving context with local contexts

A context that represents an interest group, `ambient systems`, may be spontaneously organized within the `attendees` context. At some point in time it is realized that `special interest` offers substantially better support and `ambient systems` is moved to this context. Upon its entry to the `special interest` the `ambient systems` context is informed about `attendees` already in the `special interest` (including for example `attendees` so far unaware of this initiative). Furthermore `ambient systems` is informed whenever an `attendee` (maybe another `attendee` so far unaware of this initiative) enters `special interest`.

Entity Awareness of Contexts. An entity may be aware of local contexts: Fig. 9 illustrates an entity that enters or leaves a context containing local contexts: This entity obtains information about local contexts in the context. Figure 8 illustrates entities where a context enters or leaves the enclosing context: Also here these entities obtain information about the context entering or leaving.

Upon entry of the `ambient systems` to the `special interest` any `attendee` already in this context is informed about this specific interest. Furthermore any `attendee` later on entering the context `special interest` is informed about the existence of the `ambient systems` context. The information includes the state of `ambient systems`, e.g. which events are planned and which `attendees` have registered so far for these events.

6 Experiments

Prototypical Systems. Various prototypical systems illustrate awareness of activities and contexts and that this additional form of awareness has shown to be useful and expressive.

Experiments include the ambient system Ubiquitous Doorman [12] where a user of a building is guided and supported in various activities in which the user takes part.

The user may enter and leave physical contexts in the form of various kinds of rooms and engage in activities such as meetings, lectures etc. At any time the user is supported by a helpful and informative ambient system. When walking in the building the user is immediately updated about the actual use of rooms and whereabouts of persons: Just outside a seminar room the user is informed which teaching activity takes place in the room and the state of this teaching activity including teacher, students and time information. Also guidance to specific locations and about upcoming events is supported: At any time and place the user is guided to a specific room or informed when a specific seminar begins or ends.

SoccerLab [13] is a soccer simulator designed and implemented according to a conceptual framework consisting of agents, activities, roles and contexts. Agents represent players that dynamically take on various roles in activities. Activities represent a soccer team's formation, tactics and styles as collaborative behavior of multiple players and prescribe sequences of player actions. Contexts represent environments in which players exist. Informational contexts represent among others groups of players, whereas physical contexts represent physical areas such as goal area and penalty area. When a team has the ball, the team is in its informational context `Ball Possession`. When the team loses the ball the team changes to context `No Ball Possession` in which the team immediately applies a service in order to instantiate appropriate activities e.g. `Pressure Play` and `Defend`. All players of the team engage in these activities simultaneously but their actions depend on their role on the team (defender, attacker, etc.) and the actual location of players and ball on the field (penalty area, own half, etc.). Furthermore different variants of these activities are available dependent on the actual location of the ball with respect to penalty area, goal area, or other physical contexts. Similarly the other team changes to context `Ball Possession` in which the team immediately applies a service in order to instantiate appropriate activities e.g. `Ball Control` and `Attack`. Similarly several variants of attack activities are available and the choice among these depends on the actual location of the ball and the players of the team.

Another exploratory prototype models collaboration of physical artifacts [14]. An artifact is a LEGO figure with basic structure and behavior—and models of collaborations between these artifacts are created. The models are executable—and the artifacts interact by e.g. moving around or sending messages. Children play with a family of dolls by making the dolls interact by creating and manipulating sequences of behavior. The family members engage as a group and individually in various activities e.g. social activities, work activities, sport activities. The environment includes physical contexts e.g. house, room, garden, neighborhood and informational contexts e.g. family, children, friends. Examples on awareness of activities and contexts are similar the other prototypes but the significant difference is that users are replaced by physical artifacts.

Experimental Implementation. A virtual environment in Java has three levels: The top part is a visualization of the system. The logical part at the bottom is an application framework with `Entity`, `Activity` and `Context` as abstract classes, i.e. (autonomous) entities (directing but not controlling) activities and contexts (with dynamic organization) are simulated. The simulator in the middle part maintains the

repository of entities, activities and contexts, and supports user collaboration with the virtual environment. When entities, activities and contexts move around or a user interactively moves these around, the environment visualizes the resulting awareness support.

Figure 10 gives extracts of a simple implementation of awareness notification in abstract classes `entity`, `activity` or `context` that extend abstract class `Autonomous` and implement interface `Awareness` from Fig. 11:

- Abstract method `enters(Context c, Entity e)` informs that entity `e` has entered context `c` (similarly for methods `enters(Context c, Activity a)` and `enters(Context c, Context cc)`).
- Abstract method `exits(Context c, Entity e)` informs that entity `e` is contained in context `c` (similarly for methods `exits (Context c, Activity a)` and `exits (Context c, Context cc)`).
- Method `enter(Entity e)` of abstract class `context` is invoked when entity `e` enters and notifies entities, activities and contexts in `context` about `e` (similarly for methods `enter(Activity a)` and `enter(Context c)`).

In Figs. 12 and 13 extended abstract classes `conference_Entity` and `conference_Activity` implement some *general* reaction of `enters(...)` and `exits(...)` methods. Also extended classes of `conference_Entity` and `conference_Activity` add additional, *specific* reaction to the `enters(...)` and

```
abstract class Entity extends Autonomous
      implements Awareness {
  // …
};
abstract class Activity extends Autonomous
      implements Awareness {
  // …
};
abstract class Context extends Autonomous
      implements Awareness {
  public void enter(Entity e) {
    //for each Activity a in this Context:
      a.enters(this, e);
      e.exits(this, a);
    //for each Context c in this Context:
      c.enters(this, e);
      e.exits(this, c);
  };
  public void enter(Activity a) {
    //for each Entity e in this Context:
      e.enters(this, a);
      a.exits(this, e);
  };
  public void enter(Context c) {
    //for each Entity e in this Context:
      e.enters(this, c);
      c.exits(this, e);
  };
  // …
};
```

Fig. 10. Abstract classes `Entity`, `Activity` and `Context`

```
abstract class Autonomous
                implements Runnable {
  protected abstract void Lifecycle();
  public void run()  {
    Lifecycle();
  };
  // …
};
```

```
interface Awareness {
  void enters(Context c, Entity e);
  void enters(Context c, Activity a);
  void enters(Context c, Context cc);
  void exits(Context c, Entity e);
  void exits(Context c, Activity e);
  void exits(Context c, Context cc);
};
```

Fig. 11. Abstract class `Autonomous` and interface `Awareness`

```
abstract class conference_entity
                          extends Entity {
  // …
};
class attendee extends conference_entity {
  // …
  public void enters(Context c, Context cc) {
    super.enters(c, cc);
    // …
  };
  public void exits(Context c, Context cc) {
    super.exits(c, cc);
    // …
  };
  protected void Lifecycle()  {
    //Autonomously execute requests
    //Also decide if notifications about
    //contexts are relevant
  };
};
class session_chair extends attendee {
  // …
};
class speaker extends attendee {
  // …
};
```

Fig. 12. Examples of extended classes of `Entity`

```
abstract class conference_activity
                       extends Activity {
  // …
};
class session_preparation
              extends conference_activity {
  // …
  public void enters(Context c, Entity e) {
    super.enters(c, e);
    // …
  };
  public void exits(Context c, Entity e) {
    super.exits(c, e);
    // …
  };
  protected void Lifecycle()  {
    //Directive: Forward requests
    //Also decide if notifications about
    //entities are relevant
  };
};
class session extends conference_activity {
  // …
  public void enters(Context c, Entity e) {
    super.enters(c, e);
    // …
  };
  protected void Lifecycle()  {
    //Directive: Forward requests
    //Also decide if notifications about
    //entities are relevant
  };
};
```

Fig. 13. Examples of extended classes of `Activity`

`exits(…)` methods. The figures illustrate among others extended classes for scenarios where an entity enters a context where an activity already exists (i.e. `speaker`, `seminar room`, `session`) and where an entity enters a context where a context already exist (i.e. `attendee`, `special interest`, `ambient systems`). The scenarios illustrate how the states of respectively activity and context are involved in awareness notification.

The activity scenario includes that the `session chair` and some `speakers` engaged in `session preparation` enter `meeting room`. Upon its entry the `session preparation` (through `exits(Context c, Entity e)`) is informed about `attendees` already in the `meeting room`, including `speakers` at the `session`. Also the ongoing `session preparation` is informed (through `enters(Context c, Entity e)`) whenever an `attendee` enters `meeting room`: `session preparation` can adjust to the situation so additional `speakers` get involved appropriately. In addition the activity scenario includes that an `attendee` enters a `seminar room` with an ongoing `session` and is informed about the `session` including `session chair` and `speakers` as well as the actual state including the `paper` currently being presented by the `speaker`. The scenario also includes that a missing `speaker` in the `session` suddenly enters the `seminar room` so that the ongoing `session` (through `enters(Context c, Entity`

e)) is informed about this speaker: Because session directs the sequencing of the presentations either the session can modify the schedule appropriately and directly request this speaker to make a presentation—or because session chair is also notified about the entrance of this speaker the session can request session chair to reschedule the presentations.

The context scenario includes that ambient systems enters special interest where any attendee already in this context is informed (through enters(Context c, Context cc)) about ambient systems. Furthermore any attendee later on entering special interest is informed (through exits(Context c, Context cc)) about the existence of ambient systems. An attendee notified in this way may inspect the services of ambient systems and—if appropriate—decide also to enter ambient systems.

7 Summary

Awareness of contexts and activities is an essential addition to awareness of entities: Not only entities but also ongoing activities and supportive contexts become aware of not only entities but also activities and contexts. An activity is an abstraction and the state of an ongoing activity is available: The outcome is not only an entity with methods but information about execution state and entities engaged in the activity. Also a context is an abstraction and the state of a supportive context is available: The outcome is not only time, place and services offered but information about actual entities, activities and local contexts contained in the context.

Experiments with prototypes and implementation show that awareness of activities and contexts is useful and expressive—and that simple and efficient implementation techniques are available. Challenges include how awareness of activities and contexts are integrated in existing languages, platforms and tools.

Acknowledgments. We thank Palle Nowack and Daniel May for awareness and inspiration.

References

1. May D.C.-M., Kristensen, B.B., Nowack, P.: Tangible objects: modeling in style. In: Proceedings of the Second International Conference on Generative Systems in the Electronic Arts (Second Iteration—Emergence), Australia (2001)
2. May, D.C.-M., Kristensen, B.B.: Habitats for the digitally pervasive World. In: Qvortrup, L. (ed.) Applications of Virtual Inhabited 3D Worlds. Springer, London (2004)
3. May, D.C-M.: TangO: Designing for the digitally pervasive World. Ph.D. Thesis, Maersk Mc-Kinney Moller Institute, University of Southern Denmark (2003)
4. Kristensen, B.B.: Rendezvous-based collaboration between autonomous entities: centric versus associative. Concurr. Comput. Pract. Exp. **25**(3), 289–308 (2013). Wiley Press
5. Schilit, B.N., Adams, N.I., Want, R.: Context-aware computing applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications, California. IEEE Computer Society (1994)

6. Schmidt, A., Aidoo, K.A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., Van de Velde, W.: Advanced interaction in context. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 89–101. Springer, Heidelberg (1999)
7. Agre, P.E.: Changing places: contexts of awareness in computing. Hum. Comput. Interact. **16**(2–4), 177–192 (2001)
8. Baldauf, M., Dustdar, S., Rosenberg, F.: A Survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. **2**(4), 263–277 (2007)
9. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F.A., Tanca, L.: A data-oriented survey of context models. SIGMOD Rec. **36**, 19–26 (2007)
10. Soylu, A., De Causmaecker, P., Desmet, P.: Context and adaptivity in pervasive computing environments: links with software engineering and ontological engineering. J. Softw. **4**(9), 992–1013 (2009)
11. Maclennan, B.J.: Principles of Programming Languages. Design, Evaluation and Implementation, 3rd edn. Oxford University, New York (1999)
12. Jensen, S.E.: Exploration and implementation of key aspects of ubiquitous doorman. Maersk Mc-Kinney Moller Institute, University of Southern Denmark (2005)
13. Hargesheimer, B.: Design and experiments with a general model of context-dependent collaborations between autonomous participants. Maersk Mc-Kinney Moller Institute, University of Southern Denmark (2006)
14. Kristensen, B.B., May, D., Nowack, P.: Beyond playing with physical LEGO bricks: modeling interaction between behavioral artifacts. In: IADIS International Conference on Cognition and Exploratory Learning in Digital Age, Romania (2010)