Amelie: A Recombinant Computing Framework for Ambient Awareness

Georgios Metaxas[1], Panos Markopoulos[1], and Emile Aarts[2]

[1] Eindhoven University of Technology, The Netherlands
[2] Philips research, The Netherlands
`{g.metaxas,p.markopoulos}@tue.nl,`
`emile.aarts@philips.com`

Abstract. This paper presents Amelie, a service oriented framework that supports the implementation of awareness systems. Amelie adopts the tenets of Recombinant computing to address an important non-functional requirement for Ambient Intelligence software, namely the heterogeneous combination of services and components. Amelie is founded upon FN-AAR an abstract model of Awareness Systems which enables the immediate expression and implementation of socially salient requirements, such as symmetry and social translucence. We discuss the framework and show how system behaviours can be specified using the Awareness Mark-up Language AML.

Keywords: Awareness, Focus Nimbus, Recombinant computing, Symmetry, Social Translucence, Ambient Intelligence, AML, FN-AAR.

1 Introduction

Awareness systems, can be broadly defined as systems intended to help people construct and maintain an awareness of each others' activities, context or status, even when these individuals are not co-located[4]. Awareness can bring important, if subtle, benefits, such as effectiveness of collaborative work, fostering social relationships. Awareness has been approached traditionally as a benefit deriving from the use of systems supporting cooperation between groups, messaging, and, more recently, social networking and micro-blogging.

Supporting awareness in the field of Ambient Intelligence has prompted researchers to consider how to integrate the capture of awareness information, its dissemination and display within heterogeneous collections of devices and services comprising Ambient Intelligence Environments. Example applications of such systems address well known scenarios such as awareness of a lone elderly relative living independently [10]. Existing implementations of awareness systems of this latter kind, which we call *ambient awareness systems* have so far been of very limited scale. In advancing towards realistic deployments and actual use, devices and services need to be used often in configurations and for purposes that are not foreseen by their designers and developers [8]. Eventually such a dynamic configuration and repurposing of the multitude of devices and applications in an Ambient Intelligence environment requires that they operate collectively, using information and intelligence that is hidden in the interconnection network [1]. A clear consequence of this statement is that interoperability and dynamic aggregations of devices and services are needed, a technical ambition that has been pursued consistently by the Ambient Intelligence community in the past ten years.

This paper introduces Amelie, a recombinant computing framework designed to meet this challenge. First related work is summarized and the theoretical foundations of Amelie are introduced: the Focus-Nimbus Aspects Attributes and Resources (FN-AAR) model [11], and the concept of recombinant computing [5]; then Amelie is described and the paper concludes with an illustration the advantages it offers for implementing ambient awareness systems.

2 Related Work

An early and influential model of awareness systems was the *'event propagation model'* (EPM) [14] which identifies three basic information processing functions they should support: capturing information regarding a particular individual, group, or location, disseminating it, and displaying it to the intended receivers. GroupDesk [7] is a prototype implementation of the EPM that allows users to stay informed dynamically about events, that happen currently or that have happened in the past in the surroundings of their actual position. Despite that this model did not originally target ambient intelligence it has been adopted for most implementations of ambient awareness prototypes to date, e.g., the InfoCanvas [17] and Nessie [18].

Confab [15] is a prototypical toolkit for facilitating the development of privacy-sensitive ubiquitous computing applications; building on the *Approximate Information Flows model* [16] Confab affords basic support for building ubiquitous computing applications, providing features to allow the easy implementation of a spectrum of trust and privacy levels.

3 FN-AAR

The FN-AAR model [11] is a model of awareness systems that extends the Focus/Nimbus model [2, 13] that models (the magnitude of) awareness in terms of the overlap between the focus of one entity upon another and the information that other entity is prepared to share with the first - which is termed the 'nimbus' of this latter entity. Where the focus-nimbus model describes **how much aware** two entities are about each-other in a particular *space,* FN-AAR describes **of what** are the entities **aware** in a particular situation in relation to each other, and illustrates[11, 12] how this departure is instrumental for modelling concisely awareness systems, and the social aspects pertaining to their use.

The FN-AAR model extends the focus-nimbus model, with the notions of *entities, aspects, attributes, resources and observable items.*

- ***Entities*** are representations of actors, communities, and agents.
- ***Aspects*** are any characteristics that refer to an entity's state; they are easily thought of as the complement to the statement "*I want to be aware of your…* ".

- ***Attributes*** are place-holders for the information exchanged between *Entities* by binding *aspects* with values. An entity makes its state available to other entities using attributes.
- ***Resources*** are bindings of *aspects* with ways of rendering (displaying) one or more relevant attributes. In any situation an entity might employ one or more resources to express its 'interest' about certain aspects of other entities.
- ***Observable items*** are the result of displaying some attributes about an aspect using a *resource.* Roughly speaking an *observable item* contains the answer to the question *"How are these attributes displayed to you?"*.

To reflect the fact that awareness is dynamic, the FN-AAR model populates one's ***nimbus*** with *attribute-providers*; i.e. functions that return those attributes that one makes available to other entities in a specific situation. Likewise one's *nimbus*, ***focus*** is populated with *resource-providers*; i.e. functions that return one's *resources* that display information about other entities in a specific situation. Conforming to the original *focus/nimbus* model, the negotiation of the reciprocal *foci* and *nimbi* of two entities in a given situation (i.e. the corresponding 'produced' *attributes* and *resources*) is a function which returns the *observable-items* that are displayed to the two entities about each other's states, effectively characterizing their reciprocal awareness.

4 Recombinant Computing

Edwards et al. [5, 6] pointed out that ubiquitous computing research has considered enabling technologies in isolation, e.g., location sensing, multi-device user interfaces, ad hoc network protocols, and so on, overlooking fundamental software architectural issues relating to their composition. They outline an approach that they call *"recombinant computing"* which allows the dynamic extension of computational entities through the use of mobile code. As they point out existing component frameworks (e.g., JavaBeans, DCOM, UPnP) are insufficient to enable arbitrary software interconnection because the users of such frameworks (i.e., application developers) are required not only to have knowledge of the interconnected components' interfaces but also of their semantics.

The recombinant computing approach proposes a limited set of *recombinant interfaces* that provide the foundation for allowing components to interact with one another dynamically. For that to succeed it is the users who provide the semantic understanding of what components actually do. The initial prototypical architectural framework supporting recombinant computing, Speakasy [6], outlays three general functional requirements:

- ***Connection***: How components exchange information with each other
- ***Context***: How components reveal information about themselves
- ***Control***: How components allow users and other components to effect changes

5 Recombinant Implementation of the FN-AAR

FN-AAR can serve as a conceptual model, but it can also be mapped to executable semantics following and benefiting from the recombinant computing approach discussed above.

Attribute and Resource providers (i.e. the functions that return one's focus and nimbus in a situation) are abstracted with a single recombinant interface and implemented as web services. An entity's profile comprises a set of service instances (i.e. recombinant components) that interact with one another, effectively characterizing an entity's focus-on and nimbus-to other entities. Rendering functions are abstracted as web services that implement a simple interface which provides methods for displaying information.

Amelie services communicate by exchanging attributes or resources which are the fundamental carriers of information within an awareness system; attributes and resources conform to a simple xml schema that encompasses a wide range of semantics. They are used not only to characterize effectively one's nimbus-to and focus-on others but also to characterize one's awareness of others.

The behaviour of an awareness application but also of an entity within an awareness system is defined by manipulating the services of the entity's profile. Amelie provides the necessary components that allow registration and modification of services in an entity's profile; for the same purpose services are free to provide interactive mechanisms that target any medium.

5.1 How Are the Attributes and Resources Mapped on the Architecture?

The FN-AAR model introduces the notion of attributes, and resources; the first are the elements of information exchanged between entities hence defining one's nimbus in some situation, and resources are the elements of information comprising one's focus on other entities. Bellow we will describe how these notions are mapped on Amelie.

5.1.1 Mapping Attributes

In terms of the model, an attribute is a binding of an aspect to a value. Amelie follows the model's notion of attributes through a simple XML schema. Bellow we can see an example of a simple attribute denoting that someone's activity is walking for the last hour:

```
<aml:attribute>
  <aml:aspect>activity</aml:aspect>
  <aml:value>walking for 1 hour and 25 minutes</aml:value>
</aml:attribute>
```

The value of an attribute is considered by default as a simple text. This allows heterogeneous services to display for example the attribute's information in a human-readable way. However, attribute values may be defined as structured types. Bellow we can see the same attribute value as above extended with richer semantics of a custom type.

```
<aml:value type="aml-state-duration">
 <state duration="1h25m" state="walking">
  walking for 1 hour and 25 minutes
 </state>
</aml:value>
```

The above declaration is quite richer semantically for services that are able to handle the "*aml-state-duration*" type. For example, an ambient display can benefit from the detailed semantics to make a graphical representation of the duration. On the other hand, a service that does not support the introduced value-type can still use the text part of the value only; this eliminates type-errors in the information propagation for either services, and addresses *"the tyranny of types*" one of the problems that recombinant computing is aimed at tackling.

For an entity to express its nimbus (i.e. the attributes that it exposes to others), attributes are adorned with a list of entities that they are exposed to. For example the attribute declaration below instructs the system to expose to *"John"* and *"Anna"* that our location is downtown.

```
<aml:attribute>
 <aml:aspect>location</aml:aspect>
 <aml:value>downtown</aml:value>
 <aml:access>
  <aml:entity>John</aml:entity>
  <aml:entity>Anna</aml:entity>
 </aml:access>
</aml:attribute>
```

5.1.2 Mapping Resources

Resources as described in the FN-AAR model are bindings of aspects with ways of rendering information. In terms of Amelie a resource is described with a simple XML schema, which we call Awareness Mark-up Language (AML). Bellow we can see a simple resource declaring one's focus on John's activity.

```
<aml:resource>
  <aml:entity>John</aml:entity>
  <aml:aspect>activity</aml:aspect>
  <aml:renderer target="http://home-server/picture-frame"/>
</aml: resource>
```

The above declaration instructs the system that we are acquiring John's activity, and given that he is exposing to us any information (i.e. attributes) concerning this aspect we would like to use it for displaying it using the specified renderer (in this case a "picture-frame")

In order to allow richer interaction with the rendering services the *aml: renderer* tag allows also the declaration of render-specific parameters:

```
<aml:resource>
  <aml:entity>John</aml:entity>
  <aml:aspect>activity</aml:aspect>
  <aml:renderer target="http://home-server/picture-frame">
   <color for="walking">blue</color>
     ...
   <color for="driving">green</color>
  </aml:renderer>
</aml: resource>
```

In the example above the same resource is also populated with renderer specific parameters that instruct the picture-frame to colour code different activities.

5.1.3 Enabling Seamful Design

A requirement for ambient awareness system pertains to what has been termed seamful design [3]. In most cases the 'seams' by which technological components are aggregated do not interest users and should therefore be transparent; however, the complex nature of Ambient Intelligence environments and the uncertainty that is bound to be associated with sensing and networking infrastructures, mean that it is often important to allow users to inspect and understand the nature of these seams and even to exploit them in the design of such systems. Metaxas et al. [10] argued how letting users inspect the basis of the inferences made from sensed data can let them better deal with erroneous information and can be essential for their use.

To do so, attributes may contain an optional confidence index and a section that contains information about the attribute's seams, both of which are populated by the underlying services that generate the attributes. In the above example it could be that the attribute is adorned with some seams:

```
<aml:attribute confidence="0.9">
  <aml:aspect>activity</aml:aspect>
  <aml:value>walking</aml:value>
  <aml:seams content-type="text/plain">
   the activity was detected as walking because the
   accelerometer is detecting that there is frequent
   change of the accelaration vector.
  </aml:seams>
</aml:attribute>
```

As one can observe the seams define also their content-type (above plain text) so that services can further benefit from the seams' semantic information. This is particularly useful when attributes are used to describe an entity's Observable-Items; seams allow the system not only to describe what information is displayed, but also to describe how this information is displayed (e.g. the medium, the timing, the abstraction level etc…).

5.1.4 Supporting Symmetrical Constraints

The aforementioned semantics of attributes and resources are sufficient to define one's focus and nimbus at some point in time with regards to others. At first sight, the services that generate such information are adequate protections for one's privacy considerations by choosing when, what, and to whom information should be exposed-to and when, what, and how information should be acquired-from other entities. Although this approach has been traditionally followed by relevant application-frameworks, e.g., [15] it is not sufficient to address privacy control when this is considered at a social level rather than as an issue of security or access control.

Consider for example the following scenario:

"John, an office worker seeks some distraction and a break from work. He would like to let his colleague Anna to know in case she wishes to join him. He hesitates however, as he does not want to be perceived as lazy. He would like to know Anna's mood, but would not like his interest to be known or to be public, so that he will not be perceived as prying."

This scenario above describes guardedness to disclose that characterizes many social interactions (e.g., dating, confiding) and even business transactions. Reciprocity is paramount and there is high cost at revealing intentions to others, especially in case where these are unmatched. The Amelie framework supports the application of constraints that are applied prior to the exchange of information among entities, and within the boundaries of the focus-nimbus composition, in order to support this kind of requirement regarding disclosure. In the scenario above a constraint could require the system to check whether Anna and John are sharing their feelings, before actually exposing any related information to each other.

To apply such equity constraints attributes and resources may be adorned with relevant semantic information. The attribute declaration bellow, for example, is instructing the system to expose John's mood to Anna if she is also bored

```
<aml:attribute>
 <aml:aspect>mood</aml:aspect><aml:value>bored</aml:value>
 <aml:access><aml:entity>Anna</aml:entity></aml:access>
 <aml:contstrains>
  <ctx:symmetry mode="affirmative" xmlns:ctx="…">
   <match>
    aml:attribute[aml:aspect='mood']/aml:value='bored'
   </match>
  </ctx:symmetry>
 </aml:contstrains>
</aml:attribute>
```

The above declaration uses a contextual symmetry constraint that defines an affirmative symmetrical constraint regarding sharing John's mood with Anna. The xml namespace *"xmlns: ctx"* provides the necessary executable semantics for the interpretation of the constraints; in the example above the constraints are described using XPath to define the context that needs to be matched. Similarly we could define a negative symmetrical constraint regarding the exposure of John's desire for a break to Anna, in order to reassure that John's desire for a break will only be exposed to Anna if she is not busy.

Likewise with attributes, resources may be adorned with symmetrical constraints. We could for example adorn John's resources focusing on Anna to inquire her mood only if she is focusing on his location.

Such level of symmetrical constraints both on the side of the observed and on the side of the observer support the definition of nuanced participation structures addressing privacy requirements of individuals and groups, rather than just different levels of access control.

5.2 How Are Attribute Providers, Resource Providers, and Renderers Mapped on the Architecture?

As mentioned in the introduction all the functional parts of the FN-AAR model are mapped as web services on the Amelie framework. One can imagine that an Amelie service may return attributes, when acting as an attribute-provider, resources when acting as resource-provider or even both. Similarly, attribute renderers are implemented as web services on the Amelie platform. More specifically Amelie services are implemented as SOAP (*Simple Object Access Protocol*) services over HTTP.

SOAP is a widely adapted protocol specification that is supported by all modern application development kits. Moreover its close binding to XML makes it inherently suitable to support the semantics of the XML schemes of Amelie.

In order to minimize the effort to write, configure, interconnect and use Amelie services a single WSDL (Web Services Description Language) interface is required to be implemented by all services whether acting as Attribute-providers, Resource-providers, or Render-functions. Amelie services apart from providing a core functionality (e.g. to display or return attributes and/or resources), should provide mechanisms that allow end-users to configure their instances, mechanisms that allow persistence for their instances, and methods to describe the range of output (in terms of attributes & resources) they can produce and handle.

This limited set of functional requirements along with the choice to follow well established internet standards and protocols make the development and integration of new services with in Amelie a relatively straightforward and simplified process; by allowing developers and designers of such services freely to use any toolkit, programming language, or medium that they feel more comfortable with, Amelie provides enough space to focus on their concepts.

Services that extract contextual information through wireless -sensor networks, that define and manage one's social network, that extract activities and status from one's desktop, that capture images and videos from one's context, that extract location using existing Wi-Fi infrastructure, that allow information decoration through artefacts, that allow end-users to define rules and constraints for generating meta-information, sharing and acquiring information from others, are only a small fragment of services that have been implemented on the Amelie framework addressing various end-user requirements for building up awareness systems in the past 2 years.

5.3 How Are Entities Mapped on the Architecture, What Is an Amelie Profile Like?

Each Amelie entity is assigned a profile that describes its focus-on and nimbus-to other entities in terms of the services that act as attribute and resource providers. The Amelie framework provides the necessary component infrastructure which we term Profile-Processor that handles an entity's profile semantics and produces the corresponding focus and nimbus instance in any situation (i.e. the set of attributes that the entity exposes to its contacts, and the set of resources that define what kinds of information it acquires from the rest). The structure and semantics of an Amelie profile, allows services to interconnect and cross-reference each other in order to generate meta-information and demonstrate intelligent behaviours.

Let us walk through the features of an Amelie profile, using the following simple profile that represents the focus and nimbus of some entity:

```
<aml:profile xmlns:aml="…" xmlns:inv="…">
 …
 <inv:service id="gps-location" uri="http://device.to.gps">
  <deviceid>01-234-56-789</deviceid>
 </inv:service>
 <inv:service id="myweather" uri="http://weather.forecast">
  <location>
```

```
     <inv:service ref="*"
        select="aml:attribute[aml:aspect='location']"/>
    </location>
   </inv:service>
   <inv:service id="forjohn" uri="http://expose.some.info">
     <attributes>
      <inv:service ref="myweather" select="aml:attribute"/>
     </attributes>
     <contacts>John</contacts>
   </inv:service>
   …
  </aml:profile>
```

The profile above declares three service instances, each of which defines its persistent information and its relevant service URL. The first declaration is a service instance *"gps-location"*, that instructs the profile processor that the service should be instantiated for some device with an identifier *"*01-234-56-789*"*. The <*deviceid*> is specific to the *gps-location* service, and the profile processor doesn't need to "care" about its semantic value: the profile serves as a store for the service specific parameters. The *deviceid* xml tag will be used internally by the service residing at the URL "*http://device.to.gps"* to retrieve the GPS coordinates from the specified device. Using the service declaration the profile processor prepares the connection with the service and invokes it with the provided parameters; consequently the gps-location service returns the coordinates of the device "01-234-56-789":

```
  <aml:attribute>
   <aml:aspect>location</aml:aspect>
   <aml:value type="latitude-longtitude">
    <lat value="51.4366">51 degrees,26.2 minutes North</lat>
    <long value="5.4780">5 degrees,28.7 minutes East</long>
   </aml:value>
  </aml:attribute>
```

In a similar manner the profile processor is instructed to instantiate the second service (*myweather*) with its specific parameters. We can imagine that this service implements a weather service that returns an attribute describing the weather conditions and forecast at some geo-location. Yet, notice the contents of the tag '<*location*>*':*

```
  <inv:service ref="*"
     select="aml:attribute[aml:aspect='location']"/>
```

This declaration instructs the profile processor to instantiate "*myweather"* passing to it the result of all services *(<inv:service ref="*"…)*, that return attributes regarding *"location" (aml:attribute[aml:aspect='location'])*. Given that the service "gps-location" returns a matching attribute, "*myweather"* could return an attribute such as:

```
  <aml:attribute>
   <aml:aspect>weather</aml:aspect>
   <aml:value>partly cloudy, 26° C</aml:value>
  </aml:attribute>
```

The last declaration in the profile instructs the processor to instantiate the service "*for-john*" passing to it the attributes that the service "myweather" returned previously, and an entity identifier "*John*". Once invoked, the service at *"http://expose.some.info"* could then adorn the above attributes and contact list to define that the weather information should be exposed to John.

5.3 How Is the Focus-Nimbus Composition Function Mapped on the Architecture?

The carrier of the communication objective, both in the original focus/nimbus model, and in the FN-AAR model is the focus-nimbus composition function; i.e. the function that negotiates the foci and nimbi of entities and defines the communicational outcome among them. The actual implementation of this function lies in the kernel of the Amelie framework, the Awareness Manager module. The Awareness Manager polls periodically the foci and nimbi of its registered entities pushing the appropriate attributes to the identified renderers.

For the awareness manager the profile storage of an entity and its underlying mechanisms of expressing the entity's focus and nimbus are not important. Of importance is the information each entity exposes to others (nimbus) and the information each entity acquires from other (focus) entities in a given situation. Each entity registers in the awareness manager, a service URI that identifies the entity's current focus and nimbus (i.e. its profile-processor). The awareness manager, periodically invokes the registered entities' profile-processor service URIs, and combines their instances in order to invoke the appropriate renderers according to the FN-AAR model.

Moreover, entities registered within the awareness-manager can optionally provide a URI that indicates a web service which identifies the entity's ontological model regarding the space of awareness information. This design choice not only allows an entity to observe others in its own view, but also allows the awareness manager to protect one's privacy in a more efficient way; for example it could be that Anna exposes to John that she is in the kitchen. At the same time it could be that John is focusing on Anna's activities. The ontology of John could facilitate the inference that any person who is in a kitchen is probably cooking. This way John would become aware that Anna's activity is cooking (leaving aside for the moment whether this inference would be sound or not). By having each entity register its relevant ontology, Amelie provides the required flexibility in order to use simple or more complex ontologies that relate available information to information that is needed or may be inferred. Implementation wise, third party software for defining and managing ontologies can be integrated transparently into Amelie.

The following pseudocode summarizes how the Awareness-Manager invokes the appropriate renderers while applying the focus of an entity *x*, on an entity *y*.

```
Let $f_{xy}$ = last pulled focus of x on y  (i.e. the resources that x occupies for observing y)
Let $n_{yx}$ = nimbus of y to x             (i.e. the set of attributes that y exposes to x )
Let $n_{yx}$'= apply ontology of x on $n_{yx}$  (i.e. transform y's nimbus using x's point of view)
//to apply the symmetry contstrains we need also the observer's nimbus:
Let $n_{xy}$ = nimbus of x to y (i.e. the set of attributes that y exposes to x )
Let $n_{xy}$'= apply ontology of y on $n_{xy}$
//having the reciprocal nimbi of x, and y apply the symmetry constraints
```

```
Let $n_{yx}''$=filter $n_{yx}'$ after applying constraints in relation to $n_{xy}'$
//now match the resources of $f_{xy}$ with the corresponding attributes in $n_{yx}$
For each resource r in $f_{xy}$ do
 Let a = set of attributes in $n_{yx}''$ about the aspects of the resource r
 //Invoke the renderer of the resource r passing to it the set of attributes a
 Let o = set of observable items returned from the renderer
```

Consider for example that the last known instance of John's focus at some time inquires from Anna's activity, while at the same time Anna's nimbus corresponds to an attribute that exposes to John that her location is in the kitchen. Given that John would have registered some ontology corresponding to an inference engine like the one described earlier, the Awareness-Manager would populate Anna's nimbus with the attribute that exposes to John that her activity is cooking. Consequently, and since there are not constraints that the manager needs to validate, and based on the fact that the infered attribute is exposed to John and its aspect(i.e. "activity") matches John's resource that focus on Anna, the manager would invoke the service pointed by john's resource (e.g. "*http://some.actuator.at.home*") passing to it the abovementioned attribute.

6 Conclusions

We have presented Amelie, a framework for developing Awareness Systems based on the FN-AAR model [11] and the notion of recombinant computing [5]. Amelie leaves enough space for exploratory technologies both though its openness regarding ontology engines, and through its openness through the recombinant approach. Amelie framework and AML provide the necessary semantics to directly implement systems that ensure socially salient properties for awareness systems, namely symmetry, divergent ontologies, and social translucency.

Amelie embraces desktop, pervasive, and ubiquitous services both for context-sensing and for information-decoration; the recombinant design of Amelie allows seamless and rapid integration of third-party services (they only need to support a very small set of functional requirements to complement and benefit from the framework).

Amelie respects and uplifts the designers' and end-users' role in the development of applications, allowing the first to explore the design space more easily, and the latter to control their participation and project their view on the system's intelligence.

Apart from the examples and case studies presented above the Amelie framework has been used by extensively by other research projects in the past 2 years, e.g., to support experience sampling as a research method [8], to develop and deploy a pervasive aware system [9].

Amelie's strength lies in its theoretical foundation that has been discussed extensively in this paper, its language independent implementation that allows the composition of heterogeneous components. Its focus upon modelling awareness systems provides a foundation for enabling end-users to programme awareness systems, addressing one of the challenges for Ambient Intelligence discussed in [1]. Higher level programming tools based upon Amelie are the subject of current research that aims to provide some of the power for controlling information disclosure and the social aspects of awareness directly to end-users.

Acknowledgements

This work is supported by the European Community under the "Information Society Technologies" Programme, FP6, project ASTRA IST 29266.

References

1. Aarts, E.H.L., Ruyter, B.E.R.: New research perspectives on Ambient Intelligence. J. Ambient Intelligence and Smart Environments 1, 5–14 (2009)
2. Benford, S., Fahlen, L.: A Spatial Model of Interaction in Large Virtual Environments. In: Proc. ECSCW 1993, pp. 109–124 (1993)
3. Chalmers, M., Galani, A.: Seamful Interweaving: Heterogeneity in the Theory and Design of Interactive Systems. In: Proc. DIS 2004, pp. 243–252. ACM Press, New York (2004)
4. Dourish, P., Bellotti, V.: Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work. CSCW 1992, Toronto, Ontario, Canada, November 01 - 04, pp. 107–114. ACM, New York (1992)
5. Edwards, W.K., Newman, M.W., Sedivy, J.Z.: The Case for Recombinant Computing. Tech. report CSL-01-1, Palo Alto Research Center (2001)
6. Edwards, W.K., Newman, M.W., Sedivy, J.Z., Smith, T.F., Izadi, S.: Challenge: Recombinant computing and the Speakeasy Approach. In: Proc. MobiCom 2002, pp. 279–286 (2002)
7. Fuchs, L., Pankoke-Babatz, U., Prinz, W.: Supporting cooperative awareness with local event mechanisms: The GroupDesk system. In: Proc. ECSCW 1995, pp. 247–262 (1995)
8. Khan, V., Markopoulos, P., Eggen, B., IJsselsteijn, W., de Ruyter, B.: Reconexp: a way to reduce the data loss of the experiencing sampling method. In: Proc. MobileHCI 2008, pp. 471–476. ACM, New York (2008)
9. Khan, V., Metaxas, G., Markopoulos, P.: Pervasive awareness. In: Proc. MobileHCI 2008, pp. 519–521. ACM, New York (2008)
10. Metaxas, G., Metin, B., Schneider, J.M., Markopoulos, P., de Ruyter, B.: Daily activities diarist: Supporting aging in place with semantically enriched narratives. In: Baranauskas, C., Palanque, P., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4663, pp. 390–403. Springer, Heidelberg (2007)
11. Metaxas, G., Markopoulos, P.: Aware of what? A formal model of Awareness Systems that extends the focus-nimbus model. In: Gulliksen, J., Harning, M.B., Palanque, P., van der Veer, G.C., Wesson, J. (eds.) EIS 2007. LNCS, vol. 4940, pp. 429–446. Springer, Heidelberg (2008)
12. Metaxas, G., Markopoulos, P.: Abstractions of Awareness. In: Markopoulos, P., de Ruyter, B., Mackay, W. (eds.) Awareness Systems: Advances in Theory, Methodology and Design. Springer, Heidelberg (2008)
13. Rodden, T.: Populating the Application: A Model of Awareness for Cooperative Applications. In: Proc. CSCW 1996, pp. 87–96 (1996)

14. Sohlenkamp, M., Fuchs, L., Genau, A.: Awareness and Cooperative Work: The POLITeam Approach. In: Proc. HICSS'30, pp. 549–558. IEEE Computer Society Press, Los Alamitos (1997)
15. Hong, J.I., Landay, J.A.: An architecture for privacy-sensitive ubiquitous computing. In: Proc. MobiSys 2004, pp. 177–189. ACM, New York (2004)
16. Jiang, X., Hong, J.L., Landay, J.A.: Approximate Information Flows: Socially-based Modeling of Privacy in Ubiquitous Computing. In: Borriello, G., Holmquist, L.E. (eds.) UbiComp 2002. LNCS, vol. 2498, pp. 176–193. Springer, Heidelberg (2002)
17. Miller, T., Stasko, J.: Artistically Conveying Information with the InfoCanvas. In: Proc. AVI 2002, pp. 43–50 (2002)
18. Prinz, W.: NESSIE: An Awareness Environment for Cooperative Settings. In: Proc. ECSCW 1999, pp. 391–410. Kluwer, Dordrecht (1999)