Abstract— The recent growth of social networking and sensing applications has enabled people to perform crowd computing. In turn, this activity has opened several opportunities to address people’s needs in various application areas, such as tourism, security, entertainment and emergency response. However, the design of these applications has also brought several challenges to software designers. Since users of these systems interact among them in heterogeneous physical scenarios, their interactions should be formally considered in the design of these applications in order to determine whether or not the services embedded in the system are suitable to support those interactions. The lack of guidelines to address this modeling issue not only jeopardizes the suitability of these applications, but also implies the services provided by the system can be evaluated only after implementation, which is clearly risky and expensive. This article proposes an initial modeling language that allows software designers to address this challenge through the representation of the interaction among users of a system. The interaction model represented using the proposed language allows designers to evaluate, at the design time, the information flow and the availability of interaction supporting services in the system. The usability and usefulness of the proposal are shown using a running example.

Keywords— Human-centric wireless sensor network, modeling language, crowdsensing, cooperative computing, interaction requirements, digital ecosystems

I. INTRODUCTION

The recent advances in mobile computing and wireless communication have allowed people to collaborate in several ways to reach various goals. Some computing paradigms that support this type of collaboration are not only mobile and ubiquitous computing, but also some cooperative computing approaches, such as crowdsourcing [1, 2] and crowdsensing [3] (also known as participatory sensing [4]).

Several personal activities are being supported by these computing paradigms, for instance e-business [5], tourism [6], car driver support [7] and personal security [8]. These computing paradigms can also be used to support collaborative activities, for instance in hospital work [9] and emergency response [10, 11].

Although the systems that support these new cooperative computing scenarios have shown to be useful for addressing several problems, their conception and design represent a challenge for software designers. On the one hand, there is an inherent modeling complexity that is given by the heterogeneity and dynamics of the interactions that should be supported in the application scenario. Provided that these systems are highly demanding in terms of interaction, specifying the interaction spaces in which they will be used is mandatory to determine the services that would be required by the participants to perform their activities.

On the other hand, there is no proposal to address this challenge at design time; therefore, software designers must do their best to try identifying the required supporting services, and then to evaluate their proposals once the system is implemented. This approach to conceive these systems is expensive and risky, because it is based on an uncertain set of interaction requirements.

Unfortunately, the current proposals to address the specification and analysis of these interaction scenarios provide just a partial solution, because they were conceived for other purposes. Trying to help address such a need, this paper presents an initial modeling language to represent and analyze the interactions among participants in a cooperative computing scenario. The obtained interaction model helps designers identify the services that will be required by the users to support their activities. The participants can collaborate properly only when counting on suitable services. In critical collaborative scenarios (such as emergency response), in which several types of participants must interact to reduce the impact of an event, having suitable supporting services may make a difference on the result of the response process.

The proposed modeling language is based on the proposal of Herskovic et al. [12]. Therefore, the language also allows designers to analyze the information flow among the participants. The language assumes that the collaborative systems to be modeled are structured as a Human-centric Wireless Sensor Network (HWSN) [8, 10]; therefore, the language considers the node types included in such a
specification. The usability and usefulness of the modeling language is illustrated using a running example.

Next section presents the related work. Section III briefly introduces the concept of Human-centric Wireless Sensor Network (HWSN), which is the theoretical framework supporting the modeling language. Section IV describes the modeling language. Section V presents the design of the interaction space of a personal security application, which is based on crowdsensing, as a way to show the usefulness and usability of the modeling language. Section VI presents the conclusions and future work.

II. RELATED WORK

Modeling languages may help software developers understand a process and also design an appropriate software solution. This may be especially important in complex work scenarios, like those involving mobility and collaboration. General purpose modeling tools may not be appropriate for collaborative work, as they do not represent its flexibility and variability [13], and because collaborative systems are generally multidisciplinary and technically complex in nature [14].

Several modeling languages, either created specifically for collaborative systems or adapted from mainstream use, have been proposed. Molina et al. [14] reviewed existing notations for collaborative system modeling, such as Group Task Analysis (GTA), Action Port Model (APM) and UML-G. The notations share some features that allow them, e.g. to specify groupwork, identifying roles and actors [14]. Teruel et al. [15] also compared two modeling languages: i* and CSRML. However, these approaches do not consider mobility, and the relationship between collaboration and mobility is not well understood [16]. Our previous work [12, 17] proposed a modeling language that specifically tackles collaborative and mobile issues, but not particularly in HWSN. Therefore, these modeling proposals can only partially address the design needs that come from crowd computing scenarios.

Some recent modeling approaches concern a relevant aspect in HWSN, such as context awareness, including object-role models and spatial models [18]. However, these proposals lack a view of collaborative and human aspects. Human-centric sensing presents several new challenges that need to be approached in a multidisciplinary way [19].

III. STRUCTURE OF AN HWSN

The authors introduced the concept of Human-centric Wireless Sensor Network in a previous work [10], as a way to structure the functionality that is required in these digital collaborative ecosystems. HWSN is a conceptual framework that allows software designers to characterize the role of the participants in these ecosystems. The HWSN can be represented as a bi-dimensional dynamic graph that considers the participation not only of people (interacting through computing devices), but also sensors and unmanned devices that support communication and information sharing among the participants.

The authors present a refined architecture of an HWSN in [8, 11]. It is composed of four layers (Fig.1): sensing, communication, information persistence and application. The nodes participating in an HWSN can be human-based sensors, regular sensors, mules, witness units and actuators. There are also communication units that contribute only to provide communication support in the work scenario.

Figure 1 depicts a crowdsensing application infrastructure based on an HWSN, which allows people to report and also be aware of their security condition depending on their location and the context situation information (e.g. time, weekday and light condition). The citizens collect and share security information from their surrounding environment using their mobile phones in an anonymous way. The collected information allows participants to perform an online diagnosis of their current risk and security level, while they move through urban areas [8]. Eventually, alarms (e.g. through a sound or a vibrotactile message) can be simultaneously delivered to the smartphone of user and to other nodes of the HWSN (e.g. particular neighbors or patrol officers in the area) every time that the risk level of a person overcomes a certain a pre-established threshold. Next sections describe the nodes that can be present in an HWSN, the hierarchy used to structure the services provided by those nodes, and a meta-model that summarizes the structure of these networks, which is taken as a basis for the proposed modeling language.

A. Node Types in an HWSN

As previously mentioned, five types of nodes formally participate in these networks generating, storing or sharing information to support people’s activities: human-based sensors (HBS), regular sensors (RS), mules (Mu), witness units (WU) and actuators (Ac). The HBS are people that use a (mobile) computing device to capture, share or provide information to other participants in this ecosystem. In the lower layer of an HWSN (Fig. 1) we can find all network nodes able to sense data; for instance the inhabitants of a certain area and also passersby (HBS-1 and 2), who use their
smartphone to indicate the security information of the place where they are located. In order to avoid mistakes and reduce the effort of geo-localizing that information, they use the position provided by the GPS (i.e. a regular sensor) embedded in their devices. Therefore, the HBS use their senses and also information captured from regular sensors to generate the knowledge (or meta-information) that will be shared with other network nodes; typically, other citizens and security agents.

The communication of this information among network nodes can be done using point-to-point communication (e.g. through a cellular or ad hoc network) or using mules. In the first case, communication units (CU) participate as mediators, typically antennas, which do not store or process the system information. The antennas only have the capacity to connect nodes located inside a certain communication range. The communication units are not formal network nodes, because they are not shared information holders. However, their presence affects the links status. The influence that these communication infrastructures make on the interaction capability of the users is represented through the links between nodes.

The second strategy that can be used to communicate information between network nodes involves the use of mules. A mule is a mobile node, for instance the smartphone of a passerby or a vehicle having a computer, which voluntarily offers a portion of its resources to help disseminate the information provided by other nodes. In these networks the mule could eventually have a route and a period.

In order to increase the information availability and sharing during limited communication periods, witness units (WU) can be added as network nodes. These nodes act as information gateways, allowing information sharing among temporarily uncommunicated human-based sensors. The witness units have also the capacity to fuse information (i.e. combine it in a smart way) to produce knowledge that can also be available for the network nodes. The fusion of information is done by intelligent agents that live in those units. The HBS can also serve as witness units for short time periods.

People performing crowdsensing (typically human-based sensors) use mobile collaborative applications to retrieve, update and add information to the HWSN. Therefore we can represent the status of an HWSN as follows:

\[
\text{Status (HWSN)} = f (NT, HI, LS)
\]

where \( NT \) is the network topology, \( HI \) is the information held in the network nodes, and \( LS \) are the links' status. The function \( f \) must compute the relationship among these components (according to the current links of the HWSN being evaluated) to determine availability of relevant information for the users. Information availability is a variable that we intend to maximize in this type of systems.

Actuators, which are devices able to receive an order and perform an output action, can also participate in the application layer. These devices may be a horn that emits sounds when an alarm order is received or the smartphone of a HBS that vibrates when his security level falls below a certain value. Next section briefly describes the roles of the layers considered in an HWSN and also the interlayer relationships.

### B. The Layers of an HWSN

The sensing layer is typically in charge of capturing data from the environment through regular sensors and human-based sensors. The components belonging to this layer are the only ones able to capture information from the environment and share it through the network. Information deletion is possible through timeouts (that indicate the validation period of an information piece), validity assigned by the crowd or overwriting when users' roles are utilized by the system. The information validity is dynamic, and it is established through several mechanisms, such as people’s reputation, voting, user roles and combinations of them.

The communication layer is responsible for providing an interaction capability to the network nodes. Such an interaction capability is based on the communication support provided by communication units or mules. The communication units (i.e. the communication antennas) can be fixed or mobile, which affects the status of the links between nodes. A link is always between two nodes mediated by a communication unit. These links can be permanent (represented by a continuous line in Fig. 2) or intermittent (represented by a dashed line).

![Fig. 2. Multi-layer graph representing an HWSN](image)

The mules that perform communication activities are represented as nodes of this multi-layer graph, since they are information holders. These units try to transport messages from a source to a destination node, although a path joining them may not exist. Only the destination node can process the message transported by a mule. These messages expire after a certain timeout.

The information persistence layer is responsible for storing the shared information, making it available for the HBS that are currently active in the application layer; i.e. people who are using a collaborative system on their mobile device to perform a certain activity. The participants in the information persistence layer are the HBS (playing the role of witness unit), witness units and mules. The first ones keep the shared information for short time periods, and the other node types keep the information until a timeout.

The HBS and witness units disseminate information on-demand (passive nodes) or autonomously following a predetermined period (active nodes). The active nodes are also known as beacons or information sprinklers.

Finally the application layer is responsible for providing direct and useful services to end-users, using the shared
information collected from the three lower layers. That information must flow from the sensing to the application layer, considering that the role played by a node, particularly in case of HBS, can change in short time periods.

An HBS can be active (i.e. participating as a formal network node), at the same time, in different layers of an HWSN. The presence of an HBS in a layer is directly related to the services that the node provides to others in a certain instant. Since service activation and deactivation can be done at run-time, the topology of this multi-layer network typically keeps changing most of the time. It is important to note that the topology of these networks is not related to the physical distribution and communication links among computing devices, but to the interaction capabilities among nodes through the collaborative system. Clearly there is also a network topology that is present at the routing layer (according to the OSI model), but such communication aspects will be not addressed in this article.

The eventual direct interaction among people that use these collaborative systems is supported at the application layer as shown in Figs. 1 and 2. That kind of interaction can be present or not in a system, depending on the user collaboration strategy designed to deal with the problem that such a system is trying to address.

Actuators can also be present in the application layer. Examples of these actuators are community alarms, light control systems or the mobile device used by the HBS. These actuators always make output actions, and they do not feed directly the system. Eventually, an HBS can use the output signal of an actuator (e.g. a sound alarm delivered by a horn) to create knowledge that is then input to the system. The output signals of actuators can be triggered on-demand by an HBS, or automatically when a certain situation (described by the shared data) is detected by a witness unit (through intelligent agents).

C. The HWSN Meta-model

Collaborative solutions that use an HWSN as interaction support inherit two potential benefits from these networks [10]: (1) the nodes autonomy (in terms of data and services) since these structures promote loosely-coupled computing, and (2) the information sharing can be evaluated and eventually improved at design time. However, obtaining these potential benefits requires that the interaction support platform used by the collaborative application must adhere to the structure established for an HWSN.

In order to ease the comprehension of this networked structure for software designers, we have defined an initial meta-model. This meta-model identifies the components and relationships among components that can be present in an HWSN (Fig. 3). The meta-model can also be used to evaluate on-demand if a certain system design adheres to the structure established by these networks.

The meta-model also indicates the services that every node type should provide by default. Provided that every solution is potentially different depending on the involved users and physical scenario to address, the services embedded in the meta-model should be evaluated by the system designers to determine their inclusion in the new application. These services are only those required to support information flow and interaction among the network nodes; i.e. they do not consider the particular services that every user will require to perform his/her job (also known as business services). The modeling of the business services is not part of this proposal.

Table I displays the node types supported by the meta-model, and Table II lists the supported edge types. Each node type is represented by an icon on the left-hand side of both tables, and then their meanings are described in the right-hand side of both tables.
The edges are stereotyped to indicate the level of simultaneity that the presence of two nodes has in the modeled scenario. The simultaneity attribute can assume one of three possible values: simultaneous, overlapped and disjoint. Fig. 4 indicates how this attribute will be visually represented in the interaction models.

![Simultaneity representation](image)

Fig. 4. Visual representations of simultaneity between nodes

The first one indicates that the linked nodes are active during the same time periods. In the second case, the activity periods of the nodes are overlapped, having also part of the period with only one active node. In the last case the nodes activity periods are disjoint, therefore the nodes will require an intermediary node (e.g. a witness unit) to exchange information or perform asynchronous interactions.

The intermittence of the links is determined by the type of communication support available to perform the interactions between the involved nodes. All nodes have one or more communication interfaces (i.e. antennas) that allow them to interact with other nodes. Depending on the type of interface being used by them, the communication could be permanent (e.g. a satellite network) or temporal (e.g. a mobile ad hoc network based on WiFi).

The relationship between two nodes can switch between periods of permanent and intermittent communication. In those cases the designer must assume and represent the most unfavorable situation in order to ensure that even in that situation the information will flow among them.

Concerning the network nodes, the participants can be those considered in the HWSN meta-model. All nodes have a single role, except the human-based sensors that can play various roles simultaneously. Fig. 5 shows, through an UML class diagram, the HBS default structure.

![HBS structure](image)

Fig. 5. Structure of an HBS

This modeling language does not require handling overwriting of roles, since it is assumed that an HBS can play the various roles, and each of them can be activated or deactivated on demand. Next section presents an application example that illustrates how to identify the required interaction services, based on the model of an interaction scenario.

V. APPLICATION EXAMPLE

In order to show the usefulness of the proposed language, let us consider a generic interaction scenario (i.e. a particular HWSN) in which the application for personal security described in Section III is used to try mitigating felony acts in a certain neighborhood. The participants in such a network are the neighbors (HBS), passersby (mules), patrol officers (HBS), the server of the closest police station (witness unit), the incident manager (HBS) and the operator of the 911 service.

The incident manager (i.e. a police officer) has access to surveillance cameras (regular sensors) and she coordinates the mitigation activities with the patrol officers. The neighbors can use the video-camera and GPS embedded in their smartphones (i.e. regular sensors) to report incidents and georeference the incidents information.

![Interaction model of a personal security system](image)

Fig. 6. Interaction model of a personal security system

Fig. 6 shows the design of the interaction network for such a cooperative work scenario, using the proposed language. Analyzing the mesh we observe that the only actors able to trigger a notification through the system (e.g. a warning or an alarm) are the neighbors, which is not enough. The response process could be more efficient if the 911 operator also had the capability to activate the community alarm, e.g. based on the information received from a neighbor by phone.

In the current design, the incident notification from a neighbor to the police officers must be done via the 911 operator, who must also validate and eventually route the notification to an incident manager in charge of coordinating the response actions (i.e. a police officer). The response action will be done by patrol officers. Between the neighbor who raised the notification and the officers dealing with the incident (i.e. in the critical route) there are two intermediate HBS. Since these HBS involve the action of a person, which is not automatic or predictable, we can reason that this response process will be not quick.

This kind of information can be easily provided by a software application which analyzes and evaluates the graph at design time, because the interaction model representation allows its computability. The application can also indicate situations that must be checked by the system designers. Thus, the system design can be updated incrementally according to the results of the model evaluation process. Since the model evaluation takes little time and a small effort, this process can be repeated as needed by the designers. Therefore, when the developers decide to implement the system, they already know...
its limitations (and they have the chance to fix them) in terms of interaction support and shared information flow.

The analysis of the interaction model also indicates that, e.g., the incident manager, who is a decision maker, does not have direct access to a witness unit (e.g. the police station server). Moreover, the current interaction model indicates that the passersby can eventually transport notifications from neighbors to patrol officers, but do not to other neighbors, which represents an improvement opportunity. Neighbors can deliver notifications to other neighbors only if they are simultaneously active, because they do not have support for asynchronous communication among them. All these issues represent clear limitations of the interaction model considered in the personal security system. These limitations can be automatically detected by a software application that analyzes the HWSN representing the interaction model for such a scenario. The need to count on a software application able to analyze the interaction model increases with the complexity of the graph. That application not only makes the process quick and inexpensive, but also repeatable and comprehensive.

VI. CONCLUSIONS AND FUTURE WORK

The HWSN provides a layered structure to arrange the services required to support interactions among the participants in digitally networked ecosystems, like those involved in crowdsensing. However, that proposal does not indicate which services are required to support the interaction among participants in a certain work environment. Therefore, the software designers of these solutions must do their best effort to try guessing these interaction supporting services. Furthermore, the interaction support considered in these applications can be evaluated only after the system is implemented, which is clearly expensive and risky.

This article formalizes the structure of an HWSN, through a meta-model that establishes the components that can be present in these digitally networked ecosystems. The meta-model also indicates the services that should be provided by the networked components according to the role played by them. Based on that formalization, a modeling language was proposed to represent the interaction scenario that is present in an HWSN. The modeling language is simple and allows designers to model complex interaction systems through the specification of point-to-point relationships. Since the interaction model representing a certain work environment can be specified using a computable language, the analysis and evaluation of the graph can be done automatically by a software application. This allows designers not only to evaluate their systems designs at design time, but also improve these models in an incremental way. Provided that this analysis is automatic, the process becomes repeatable, quick and inexpensive. These features give to the proposed language a comparative advantage over other available proposals.

The usefulness of the proposed language was illustrated with a brief application example. The usefulness of this modeling and analysis process increases with the complexity of the system.

There are some mechanisms to protect the shared information and users’ identity (i.e. security and privacy issues) that were not considered in the proposed language, and therefore they will be addressed as future work.

VII. ACKNOWLEDGMENTS

The work of Alvaro Monares has been supported by the Ph.D. Scholarship Program of Conicyt Chile (CONICYT-PCHA/Doctorado Nacional/2010-21100285). This work has also been partially supported by Fondecyt (Chile), grants: 1120207 and 11110056.

VIII. REFERENCES