Social Computing Networks: A New Paradigm for Engineering Self-Adaptive Pervasive Software Systems

Naeem Esfahani
Department of Computer Science
George Mason University
nesfaha2@gmu.edu

Sam Malek
Department of Computer Science
George Mason University
smalek@gmu.edu

1. INTRODUCTION

Software systems are increasingly permeating a variety of domains including medical, industrial automation, and emergency response. The advances in portable and embedded computing devices and the recent advances in wireless network connectivity have paved the way for the proliferation of *smart spaces* in such domains. At the same time, the emergence of service-oriented technology (e.g., web services [10]) and interoperability standards (e.g., WSDL [11], UDDI [5]) has made it possible to develop pervasive software systems intended for execution in smart spaces that were not even conceivable a few years back.

As the boundary between cyber and physical systems blurs, domain experts and end-users increasingly rely on such systems for their day to day activities. The software deployed in such settings needs to deal with the reality that pervasive environments are innately dynamic and unpredictable. Moreover, the users' requirements for such software systems are often not completely known at design-time, and even if they are, they may change at runtime. These characteristics have forced the engineers of such systems to deal with two emerging and increasingly important requirements: (1) rapid composition of software systems at runtime based on the user's changing needs, and (2) autonomous adaptation of the software system at runtime to satisfy the system's functional and quality of service (QoS) requirements.

Unfortunately, engineers have faced major obstacles with automatic composition and adaptation of software systems using *pervasive computing resources*. A *resource* abstractly refers to the contemporary computing entities, software or hardware, such as portable devices, sensors, software services, smart homes, and etc. These obstacles are rooted in the observation that automatic composition of pervasive software hinges on the availability of a *semantic knowledge* among the resources, e.g., the ability to understand the relationship between the different elements of a system, infer the intricate behavior of users, know which components and users can be trusted, where to look for particular resources, when to initiate change in the system, and so on.

In parallel with the advancements in smart spaces and pervasive software systems, and largely unaffected by them, we have witnessed the growing popularity of *online social networks* (we will refer them as simply social networks henceforth). Social networks provide a common interaction infrastructure (e.g., instant messaging, creation of user profiles, linking of profiles) that are used to grow online communities of people with shared interests and activities. It is commonly known that social networks embody a large body of knowledge that may be used for a variety of purposes, such as targeted marketing of products, and identification of suspicious and criminal activities.

An underlying insight guiding our research is that by harnessing the knowledge embedded in social networks, one could significantly improve the manner in which pervasive software systems are developed and deployed. In this paper, we introduce a new computing paradigm, entitled *Social Computing Networks (SCN)*, which uses the intelligence (semantic knowledge) embedded in social networks along with the recent advances in end-user software engineering, automatic software composition, and application interoperability standards to radically change the manner in which pervasive software systems are constructed.

The proposed paradigm is realized via an integrated framework consisting of (1) an extended social network structure that enables sharing, discovery, and utilization of pervasive resources within a trusted group of individuals, (2) an intuitive visual language that can be used for expressing a pervasive system's requirements in terms of resources available within a user's social network, (3) an infrastructure for socially-aware discovery of suitable resources and composition of software systems, and (4) runtime monitoring and adaptation of the software in response to observed changes in the context. The remainder of this paper provides an overview of the general paradigm and our ongoing efforts in realizing it.

2. MOTIVATING SCENARIO

We rely on a simple medical emergency scenario to illustrate the various concepts in our research. Consider that one method of decreasing medical costs is to release patients from the hospital as soon as possible. However, some patients need further supervision after they are released. For instance, heart rate of a patient with a history of cardiac arrest needs to be monitored, and depending on the nature of emergency appropriate remediating actions need to take place. Unfortunately, state of the art technologies currently employed in dealing with such situations are unwieldy, ad-hoc, and manual. For example, a heart rate monitor device may be used to alarm an emergency, but informing the appropriate relatives, engaging the appropriate doctor, and potentially resolving the situation is largely performed manually.

Figure 1 depicts a concrete scenario, where Agnes has just been moved to home after a surgery under supervision of Dr. Ernest. Her children, Annette and Oliver, are living nearby and usually look after her when she is sick. In such setting, Agnes may want to report her heart readings to the doctor. She may also want to send an alert both to her doctor and one of her children when a

ICSE '10, May 2-8, 2010, Cape Town, South Africa

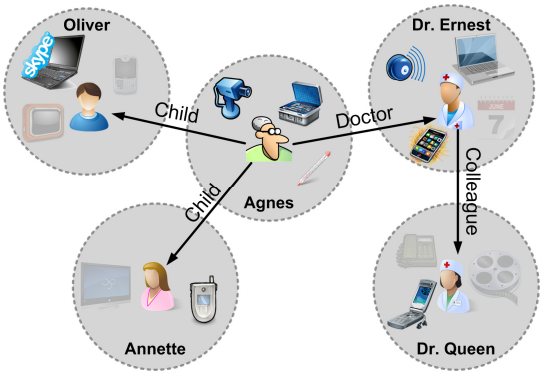

Figure 1. An example of extended social networks: relationships represent a subset of Agnes's turf network.

dramatic change is detected in her heart rate (e.g., above 120 bpm). A fully automated solution to this scenario could involve Agnes's heart monitor as the sensor, both her children's and Dr. Ernest's alert system for the notification, and finally the servers at Dr. Ernest's hospital for recording the heart readings.

The key underlying insight is that while the hardware and network capabilities required to fully automate such a simple pervasive system already exist, it is the software engineering issues, in particular those of trust and privacy, that make the realization of such systems difficult. This is precisely the focus of our research.

3. SOCIAL COMPUTING NETWORKS

Social Computing Networks (SCN) is a new paradigm for automatically composing pervasive software systems that draws heavily on the knowledge provisioned by social networks. Unlike traditional software composition methods, SCN targets end-users as well as software engineers. Figure 2 provides an overview of the SCN framework. In the following subsections, we describe the various components of SCN in detail.

3.1 Extended Social Networks

We propose a reconceptualization of social networks, where the notion of an individual is expanded to include that person's smart surroundings, which we call *turf*. An individual's turf consists of a variety of cyber-physical resources, such as smart mobile platforms (e.g., cell phone, PDA), sensors and actuators in a smart building (e.g., smoke detector, fire sprinkler), software services (e.g., a digital calendar), and wearable devices (e.g., heart monitor). We refer to this broader notion of social networks as *extended social networks (ESN)*.

Individuals in ESN are not only affiliated with one another, but also with resources provisioned by their respective turfs, which we call *turf network*. An individual's turf network consists of resources available in one's turf, and potentially some of the resources available in the affiliates' turf, and recursively their neighbor's turf. A user explicitly specifies (1) which resources from her turf are made available to her direct affiliates, and (2) the scope of sharing for each resource, i.e., whether it could be provisioned to indirectly connected affiliates. Note that access to a resource may also be granted to a *group*, which similar to traditional social networks is a classification of one's affiliates (e.g., children, parents, family friends, coworkers at a company, and medical clinic staff). As further elaborated on below, turf networks lay the foundation for ensuring and enforcing *trust* and *privacy* in the framework.

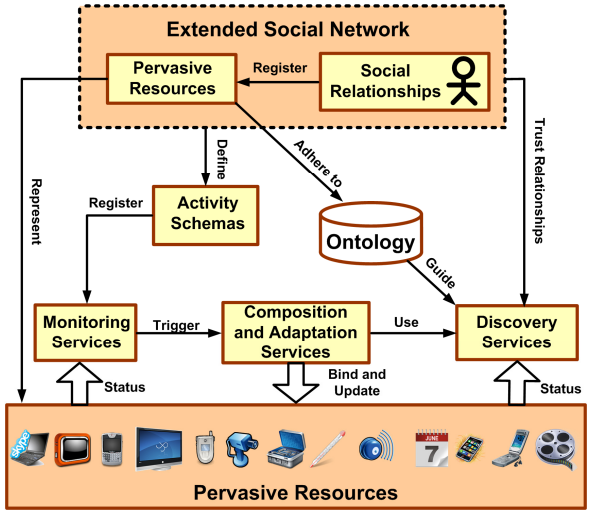

Figure 2. The high-level structural view of Social Computing Networks framework.

Figure 1 shows an ESN that corresponds to the example of Section 2. Each circle defines a user's turf. Users may incorporate various pervasive resources (e.g., smart devices, software services) into their turf. For example, Agnes has already registered three devices with her turf: camera, heart monitor, and digital thermometer. The items that are not faded in the turfs correspond to resources that Agnes may access, and constitute her turf network. Annette has decided that her parent can access her cell phone, but not her TV. Similarly, Oliver has given his parents access to his Skype, but not to his PDA and TV. Dr. Ernest has granted her patients permissions to interact with a logging service running in his office server and cell-phone, but not his calendar. Finally, Drs. Queen and Ernest, who are colleagues at the same clinic, have granted permission to one another's cell phones under emergency scenarios.

There are two underlying assumptions in our research (1) both logical and physical resources are selectively made available within one's turf network via a software interface, referred to as a *service*; and (2) the resources are compliant to a corresponding service type specification in a *domain ontology*. The ontology consists of a set of terms, concepts, and definitions that allow one to precisely specify the services and capabilities of entities within a domain. For instance, it specifies how the services provided by a pervasive resource can be accessed. Hardware devices and software services are built to conform to these standards. In a sense, the role of ontology in SCN is analogues to that of a device driver in an operating system. The manufacturers of the devices and services usually form these standards (e.g., [9]) and build the functionality according to them.

We believe the aforementioned vision of social networks is not only feasible, but already well underway. In fact, some existing social network environments, such as Facebook and Google wave, already provide integration support with various devices (e.g., Android cell phones) as well as various software services (e.g., Google calendar). In turn, they enable selective sharing of such capabilities with individuals from one's network.

3.2 Social Activity Schemas

As mentioned before, in the pervasive setting, the users' requirements are often not completely known at design-time, and even if they are, they may change at runtime. To that end, we have

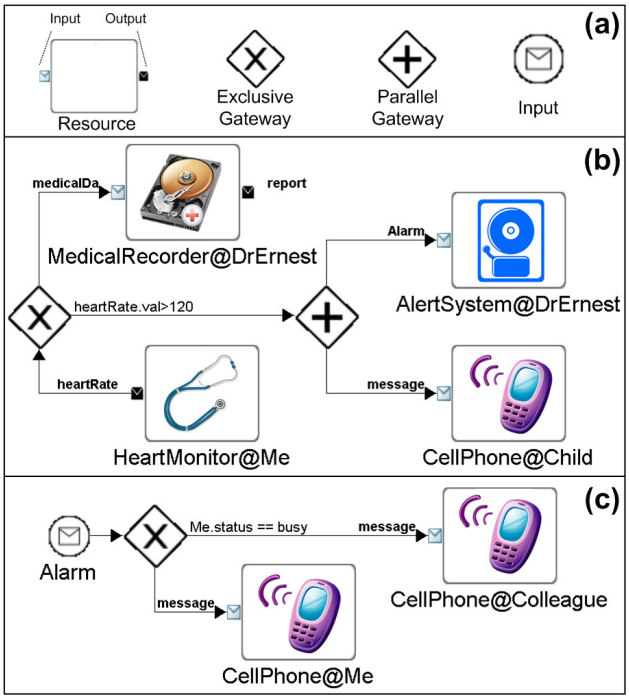

Figure 3. Using SAS for specifying a particular use case: a) language constructs, b) Agnes's heart monitoring use case, and c) Dr. Ernest's specification of *AlertSystem*.

developed an intuitive modeling language, called *Social Activity Schemas (SAS)*. As shown in Figure 3, SAS is intended to be used by ESN users to visually specify the system's requirements in terms of the resources in their turf network. For brevity we describe only a small subset of SAS, which builds on a software composition language developed in our recent work [3], and enriches that with the constructs needed in the SCN paradigm.

Figure 3a shows a small subset of the language constructs. These constructs are connected together using *Flows* to denote the sequencing. *Resource* models a logical (e.g., software service) or physical (e.g., a sensor potentially made available to the rest of the system via a software wrapper) entity and consists of several *Input* and *Output* ports. The name of a resource is composed of two parts divided by "@"; the first part indicates the type of the resource, while the second part designates a set of turfs from which the provider of the resource can be selected. The resource type is defined in a domain ontology. As detailed in Section 3.4, resolving the concrete provider at runtime depends on the relationships established in the ESN. *Gateways* manage the flow of control within a scenario. *Exclusive Gateway* works as an "or" switch, which selects a subset of outgoing flows based on a condition. *Parallel Gateway* on the other hand is similar to an "and" fork, which enables all the outgoing flows.

Figure 3b shows the requirements specified by Agnes for the example of Section 2 using SAS. As depicted in Figure 3b, the use case starts with the *HeartMonitor* belonging to Agnes. The *HeartMonitor* sends the *heartRate* readings to an *Exclusive Gateway*. The *Exclusive Gateway* detects if the *heartRate* value is more than 120, which is considered dangerous. If so, using a *Parallel Gateway* a message is sent to the *Alert System* of *Dr. Ernest* and *CellPhone* of a *Child*. Otherwise, a message is sent to *Dr. Ernest's MedicalRecorder*, which records the readings for future diagnosis. As will be detailed in the following sections, usually it is sufficient to find only one instance of each resource that satisfies the social relationship. For example, for *"CellPhone@Child"* it is enough to find an active *CellPhone* belonging to one of Agnes's children (in this case Annette).

An individual may share an SAS with its affiliates by exposing its *Input* and *Output* ports. For instance, Figure 3c shows Dr. Ernest's specification of *AlertSystem*, which as you may recall was used by Agnes in Figure 3b. When an *Alarm* arrives, an appropriate *message* is sent to his *CellPhone*; unless he is busy, in which case the *message* is routed to the *CellPhone* of an available *Colleag*ue working at the same hospital (e.g., Dr. Queen's). Note that *Me* refers to the person defining the SAS.

3.3 Monitoring

Every SAS model may have some preconditions that determine when it should be executed. For example, only after *HeartMonitor* is attached to Agnes, the scenario of Figure 3b should be initiated. Therefore, as shown in Figure 2 after an SAS is defined, it is registered with SCN monitoring services. Note that monitoring is not limited to pre-deployment phase; since pervasive systems are inherently dynamic and unpredictable, some changes may happen in the environment causing the need for change in deployed SAS. The composition and adaptation services are in turn triggered to handle the detected changes.

3.4 Discovery

Recall that an end-user could specify a resource in two ways: (1) fully specified, e.g., AlertSystem@DrErnest, and (2) group specified, e.g., CellPhone@Child. Groups allow the resulting system to be significantly more flexible, as further elaborated on below. For deploying SAS models, a discovery mechanism is needed that based on the relationships within the social network resolves the resources types (placeholders) with the actual providers (instances). Clearly, when a resource is fully specified, the discovery is trivial. On the contrary, when a resource is group specified, the discovery needs to take the dynamic nature of turf networks into account. For instance, some of the resources within one's turf network may not be available. You may also recall that the scope of an individual's turf network may expand several levels, i.e., beyond one's immediate affiliates. Hence, the discovery mechanism needs to take into account accessibility, trust, and availability of the resources within one's turf network.

For each resource, the search is performed in three steps. Consider *CellPhone@Child* in Figure 3b: first, turfs satisfying the *Child* relationship with Agnes (i.e., Annette and Oliver) are selected; afterwards, the resources within those turfs are narrowed down to those that comply with the *CellPhone* specification from the domain ontology; finally, the complying resources to which Agnes has access are selected (recall from Section 3.1 that access control defines Agnes's turf network). In this example the two resources would be Annette's cell-phone and Oliver's Skype. The discovery mechanism then selects one of them based on an end-user configurable policy (e.g., randomly, priority, user discretion).

3.5 Composition and Adaptation

As depicted in Figure 2, once the preconditions for a given schema are satisfied, the composition is initiated. The resources for that schema are discovered, selected, and bound to one another to compose the running software system. A software system constructed in this manner may itself be exposed as a resource to affiliates, and reused in composing more complex pervasive

software systems (e.g., *AlertSystem* in Figure 3). Finally, the resources in the composed software system are coordinated using an SCN infrastructure engine that executes the SAS model.

As mentioned before in Section 3.3, some changes in the environment may impact the system's functional or QoS properties. If such changes are detected, the adaptation services work to alleviate the situation as follows:

- *A precondition is no longer valid.* For example, when Agnes removes the heart monitor. In such cases, the executing SAS model is stopped, resources are unbound, and the model is registered for monitoring again.
- *One of the resources discovered previously is no longer available.* In this case, another online discovery is performed to find a substitute resource that adheres to social relationships and resource types. For example, consider when Annette's cell-phone runs out of battery. In this case, Oliver's Skype provides a suitable alternative, since it satisfies the social relationship (i.e., *Child*), service type specification (i.e., *CellPhone*), and trust relationships (i.e, it is included in Agnes's turf network).

4. FRAMEWORK PROTOTYPE

We are developing an initial prototype of the SCN framework on top of existing standards and open social network frameworks, which we believe aid its wide-scale adoption. Our prototype complies with the web services standards. We are relying on UDDI [5] for the implementation of the domain ontology, and WSDL [11] for service description and discovery. We are also targeting domains, such as emergency response, that already have well established ontologies (e.g., [9]).

To realize the ESN component of our framework (recall Section 3.1), we are building on top of the Google Wave technology [8]. Google Wave is an open-source social networking environment that provides a number of advanced facilities. Google Wave provides support for real-time event broadcasting, which we are leveraging for exchange of messages among the pervasive resources. Google Wave also provides *robots* and *gadgets*, which provide a way of cutomizing its services.

Gadget allows for extending Wave's application environment, which we have used to support various SCN concepts. For example, our visual SAS editor is built as a gadget, which enables the users to construct a SAS model by dragging and dropping resources available within their turf network. *Robot* is an automated agent deployed on a Wave server to extend its capabilities. Within ESN, each pervasive resource is realized as a robot, which essentially acts as a software wrapper to enable interaction with the resource. Each robot complies with the specification of a resource type from the domain ontology.

5. RELATED WORK

Numerous frameworks and technologies for pervasive computing have been proposed (e.g., [1,2,4,6]). However, with the exception of [1,6], none of the existing solutions employ social networks. Researchers in Socialnets [6] have tried to build a framework to establish direct trust relationships between handheld devices based on social relationships among the people. These relationships allow disseminating information without the need to maintain end-to-end connectivity between devices. On the other hand, the goal of ASTRA project [1] is to come up with a framework based on pervasive awareness to assist social relationships. This framework facilitates spreading information about people to help them maintain tight social connections. None of these works, however, aim to provide the ability to dynamically compose and adapt a software system. Similar to our activity oriented language, uDesign [7] targets smart patient care spaces. However, its focus has not been on incorporating social networks.

6. CONCLUDING REMARKS

We presented a new paradigm that relies on social networks as well as automated software composition techniques to alleviate some of the challenges of constructing pervasive software systems. We are proposing a reconceptualization of social networks that beyond simple human interaction, enable sharing, discovery, and utilization of pervasive computing resources. Pervasive services and smart devices in our approach mimic the social relationships among the people, and not only enable flexible and efficient discovery of resources, but also alleviate trust and privacy concerns.

Avenues of future work involve fully implementing and evaluating the approach in real-world settings. We are also exploring a method of *inferring* trust between individuals when they are not directly affiliated. Currently trust is established *explicitly* based on the user's existing relationships. However, even if individuals are not directly connected, they may be able to establish trust with one another, albeit at a lower threshold. In the example of Figure 1, while Oliver and Dr. Ernest are not directly related, their mutual relationship with Agnes allows both parties to conclude a higher level of trust than they would with a stranger.

7. ACKNOWLEDGMENTS

This work is partially supported by grant CCF-0820060 from the National Science Foundation.

8. REFERENCES

[1] ASTRA Project, http://www.astra-project.net/.

[2] Chetan, S., et al. 2005 Mobile Gaia: a middleware for ad-hoc pervasive computing. Proc. of IEEE Consumer Communications and Networking Conference, 2005.

[3] Esfahani, N., Malek, S., et al. 2009 A Modeling Language for Activity-Oriented Composition of Service-Oriented Software Systems. Model Driven Engineering Languages and Systems, Denver, Colorado, USA: 2009, pp. 591-605.

[4] Garlan, D., et al. 2002 Project Aura: Toward Distraction-Free Pervasive Computing. IEEE Pervasive Computing, vol. 1, 2002, pp. 22-31.

[5] Papazoglou, M. 2007 Web Services: Principles and Technology, Prentice Hall, 2007.

[6] Socialnets, http://www.social-nets.eu/.

[7] Sousa, J.P., et al. 2008 uDesign: End-User Design Applied to Monitoring and Control Applications for Smart Spaces. Working Conference on Software Architecture, 2008.

[8] Trapani, G. and Pash, A. 2010 The Complete Guide to Google Wave. http://completewaveguide.com/.

[9] US GOV. US Government Web Services and XML Data Sources. http://www.usgovxml.com/.

[10] W3C. Web Services. http://www.w3.org/2002/ws/.

[11] Weerawarana, S., et al. Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall, 2005.