Empowering End Users for Social Internet of Things

Ji Eun Kim
Bosch Research and Technology Center
Pittsburgh, PA, USA
jieun.kim@us.bosch.com

Xiangmin Fan
University of Pittsburgh
Pittsburgh, PA, USA
xiangmin@cs.pitt.edu

Daniel Mosse
University of Pittsburgh
Pittsburgh, PA, USA
mosse@cs.pitt.edu

ABSTRACT

We present *Socialite*, a novel end user programming tool for the *Social Internet of Things* (SIoT). SIoT is a new paradigm where IoT merges with social networks, allowing people and connected devices as well as the devices themselves to interact within a social network framework. Through an online survey with 60 potential users, we identified eight desired features for the SIoT, which were then clustered into four rule categories that can be programmed by end users and/or imposed by systems. The rules created by end users are used to reason about both devices and people in their social relationships to support automated decisions during runtime. *Socialite* uses ontology/semantic models for basic/low-level knowledge representation (e.g., device and user) to encapsulate the heterogeneity in devices from various manufacturers, and uses production rules (trigger-action programming) for high-level reasoning. With the ontology model, our reasoning supports both device type automation (e.g., current temperature from a thermostat) and capability-based automation (e.g., current temperature from any devices with the same capability). Furthermore, the *Socialite* rules leverage social relationships and device capabilities to facilitate collaboration by efficiently sharing configuration and information among users/friends and even with devices from people unknown to a user. In a 24-participant user study (12 with no programming experience), we found that *Socialite* was easy to learn and use, for both programmers and non-programmers. Participants were able to create automation based rules, social relationship involved rules, as well rules they created during the study.

CCS CONCEPTS

•Human-centered computing →Human computer interaction (HCI); *Collaborative and social computing;* **•Computer systems organization →***Embedded and cyber-physical systems;*

KEYWORDS

Social Internet of Things, End User Programming, Semantic Reasoning Framework

ACM Reference format:
Ji Eun Kim, Xiangmin Fan, and Daniel Mosse. 2017. Empowering End Users for Social Internet of Things. In *Proceedings of The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA USA, April 2017 (IoTDI 2017),* 12 pages.
DOI: 10.1145/3054977.3054987

IoTDI 2017, Pittsburgh, PA USA

DOI: 10.1145/3054977.3054987

1 INTRODUCTION

By merging the Internet of Things with social networks, a new paradigm called Social Internet of Things (SIoT) is created [7, 22, 30, 31]. SIoT allows people and connected devices, as well as devices themselves, to interact in a social network framework. The information generated from devices and people will become sharable (to the extent access control allows it) by SIoT participants [22]. We create a new system to instantiate the SIoT, called *Socialite* that enables users and devices in SIoT to reinvent *social navigation* [15, 16, 28], that is, the discovery and sharing of information, by leveraging new social relationships. For example, a *device* encounters an error and finds likely repair solutions based on its relationships *with other devices* that experienced the same error.

We anticipate a new relationship revolution will be enabled by the SIoT with the collaboration of people and devices through these new social relationships. Being an extension of social networks, the SIoT can increase *social capital* further and create opportunities to improve quality of life in the era of the IoT. Social capital [2, 11, 25] is what provides access to resources embedded in social relationships and enables people to mobilise these embedded resources to facilitate action. More specifically, users get the ability to control their devices, activate services, and contribute to common goals (e.g., saving energy, securing homes in a town, or fixing a device automatically through peers) by leveraging the shared information obtained through the SIoT.

While automated actions are enabled in SIoT, the decision in a certain situation during runtime should be personalized for each individual user. To support personalization in SIoT, *Socialite* is a novel end user programming tool that empowers end users to create their own rules (i.e., applications). End user rules are used to reason about both devices, people, and their relationships, and to enable the system to make automated decisions when triggering these rules.

We conducted an online user survey with 60 participants to identify the important and desirable features of the SIoT. The features analyzed from the online survey were classified into rule categories to develop the *Socialite* reasoning concept. *Socialite* uses ontology/semantic models for basic/low-level knowledge representation (e.g., device and user) to encapsulate the heterogeneity in devices from various manufacturers, and uses production rules (trigger-action programming) for high-level reasoning. Since our low-level knowledge is based on the ontology model, our reasoning supports two types of automation: (a) device type (e.g., get the current

temperature from a thermostat) and (b) capability-based (e.g., get the current temperature from any devices with the same capability) automation. Furthermore, the *Socialite* rules leverage social relationships and device capabilities to facilitate collaboration by efficiently sharing configuration and information among users and even with devices from people unknown to a user.

Inspired by the design guidelines [12] for end user programming and our survey results, we present the design, prototype, and evaluation of a web based trigger-action style visual programming tool for end users to create their own rules within the *Socialite* reasoning framework. We implemented this reasoning concept in a system that integrates off-the-shelf devices as well as virtual devices for simulation of real/more devices.

Through a 24-participant (including 12 without programming experience) user study with our system, we found that *Socialite* interface was easy to learn and use, even for participants with no programming experience. The participants were able to use *Socialite* to create automation-based rules, social relationship involved rules, as well as rules that they created during the study, after spending minimal efforts on practice. This clearly empowers users to interact with the IoT, instantiating and realizing the SIoT paradigm.

The remaining of the paper is organized as follows: The related work is discussed in Section 2. The *Socialite* system overview is explained in Section 3 followed by the discussion of new social relationships and applications in Section 4. Section 5 presents our online survey methods and the data analysis. Section 6 illustrates the *Socialite* reasoning framework. The user study of the *Socialite* system is presented in Section 7. Section 8 concludes our work.

2 RELATED WORK

In the early phase of the Internet of Things (IoT), researchers [9, 19] started to integrate existing social networking sites as media for publishing and sharing the data from the connected devices. Recently the convergence of the IoT and social networks has been proposed as a promising direction for the future IoT systems [5–7, 30, 31]. A few platforms such as Paraimpu [32] and SenseWeb [21] have been proposed to allow sharing of the objects with other users using Web services to send the data to a central server. However, those approaches do not include different social relationships thus the potential collaboration of users and devices is limited.

In [27], social devices for co-located devices and humans were introduced to interact with each other. Their approaches aimed to enhance the remote communication for the social media services when users and devices are located in the same space. Similar to our social relationship models, the work in [6] devised different relationships between human and devices. Although it discusses the high-level conceptual architecture model including the core components of the system (e.g., ID management, service discovery and composition, and trustworthiness management), the concept has not been realized in a real system.

Empowering users to program their smart environment has been discussed in academic literature for decades. The authors in [8] reported that autonomous technologies often make users feel out of control when they are not able to adjust the level of autonomy of their home according to their needs. The author in [29] argued that end user configurability is crucial in smart home applications

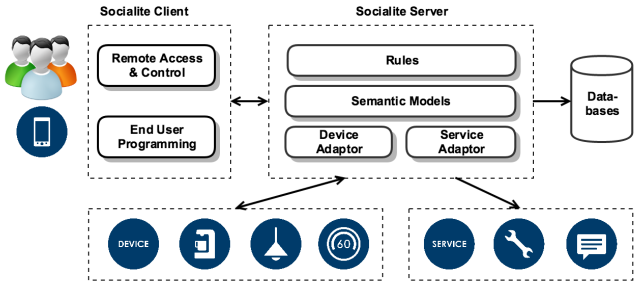

Figure 1: *Socialite* system overview

and advocated sharing insights across a community of users. The authors in [12] analyzed 47 papers from ACM and IEEE Xplore Digital Libraries for end user programming, and proposed the set of guidelines recommending trigger-action and trigger-constraint-action formats for representing smart home rules.

The real-world end user programming tools incorporating connected devices and web services have become popular over the last few years [36]. IFTTT [20], Atooma [4] and WigWag [38] are examples in industry.

However, the current solutions are limited to rules for a single device type rather than rules for each device capability. Furthermore, given the novelty of the SIoT, social relationships are not supported in end user programming and the underlying reasoning engines. To the best of our knowledge, our online survey and the end user programming prototype supported by a scalable reasoning engine are the first research contribution that incorporate various categories of rules for SIoT in smart home applications.

3 SOCIALITE SYSTEM OVERVIEW

Figure 1 illustrates the high level overview of the *Socialite* System, comprising a web based *Socialite Client* application, a distributed *Socialite Server* accessing devices and services and multiple *Databases*.

The users of *Socialite* can *remotely access and control* their connected devices using the *Socialite Client* application. Our earlier client application [22] supported the basic functions including the management of users' devices and relationships, and remote access and control of connected devices in relationships. In this paper, we enhance our application by enabling users to create their own rules (e.g., if my friend is in my living room, set the thermostat temperature to my friend's preference value) through the *end user programming* user interface.

The *Socialite Server* provides a uniform access to heterogeneous devices made by different manufacturers by decoupling the common data models for devices (represented in *Semantic Model*) and the manufacturer specific device implementations in *Device Adaptor*. Similarly, any REST services provided by third parties are uniformly accessible through *Semantic Model*, which is decoupled from the implementation of *Service Adaptor*.

The *Socialite Semantic Models* (see Fig 2) represent the user and device information together with its history (e.g., attempted repair solutions), static and dynamic locations of users and devices, services and relationships between users and devices. The model

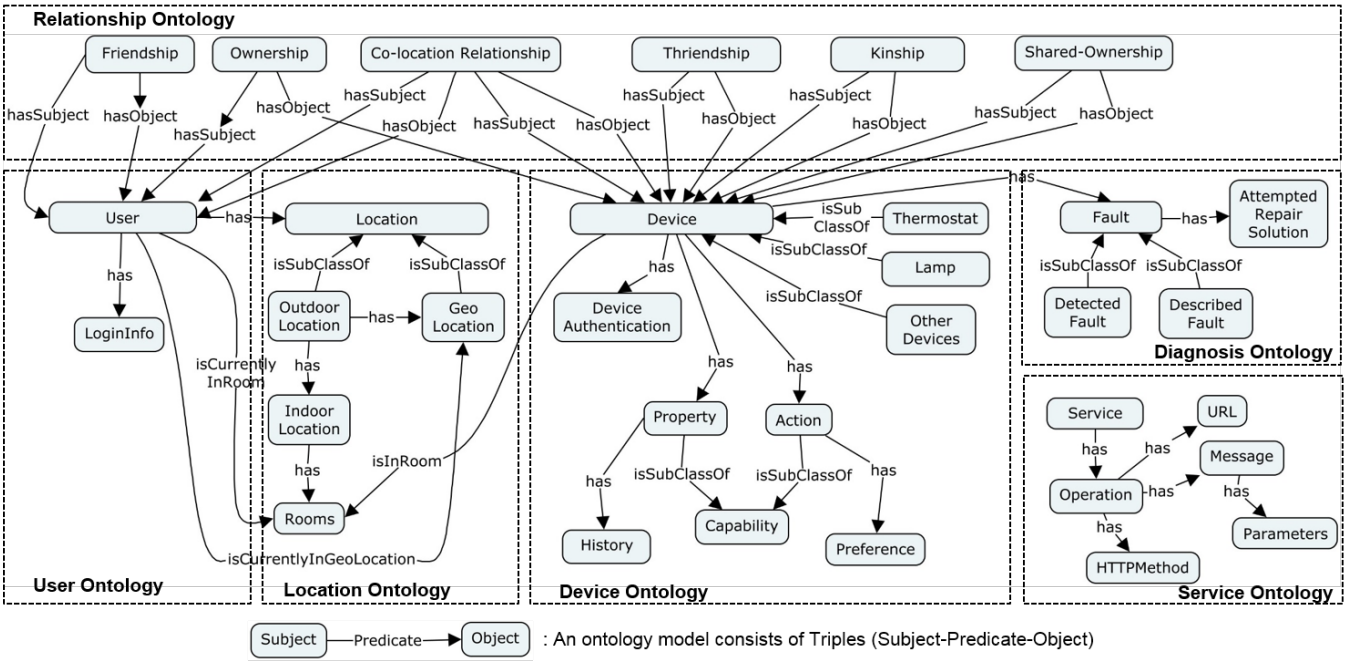

Figure 2: Graphical representation of the core ontologies in *Socialite* (Shortend)

presented here enhances the work in [22] with diagnostic and service ontologies to support all features identified from the user study (see Section 5). The *Rules* created by end users run in the reasoning engine to make an automated decision upon an event (e.g., device status change, user's location change) during runtime.

The *Databases* manage persistent data including storing the data for semantic models, device histories and rules.

4 NEW SOCIAL RELATIONSHIPS AND APPLICATIONS

The benefits of the *Socialite* system become greater when the new relationships are established for exchanging, distributing and receiving the data generated by the users and devices, and these relationships contribute social navigation that facilitate use by other devices and users.

We defined new relationship types between people and devices in our earlier work [22]. This section summarizes the new social relationships for SIoT and presents envisioned applications that leverage the new social relationships.

4.1 New Social Relationships (from [22])

Socialite defines new relationship types (see Table 1) that allow humans and devices in the SIoT to collaborate, complementing existing social network *friendships*.

Users and devices participate in *ownership* if a user registers a device in *Socialite*. Users and devices have location information, and the *co-location relationship* between them is dynamically updated, for example when their locations are changed.

Three new *Socialite* relationships between devices are (a) *kinship* for the same model of devices from the same manufacturer, (b) *thriendship* (things of friends) for the relationship between devices owned by friends and (c) *shared ownership* for the devices owned by the same user.

Table 1: New relationship types for SIoT from [22]

Relationship type	Relationship definition
Friendship	Relationship between users, as in social networks
Ownership	A device registered by its owner
Co-location	Users and/or devices in the same location
Kinship	Devices with the same model and manufacturer
Thriendship	Friendship among things/devices of friends
Shared Ownership	Devices owned by the same user

4.2 Envisioned Applications Built on New Social Relationships

We present three envisioned applications to illustrate how users could benefit from adopting *Socialite* to support the new relationships in SIoT for sharing information with other people and devices.

Device Life-cycle (Configuration, Operation and Repair) Management

Devices involved in the *kinship* relationship can improve their performance by sharing parameters and configurations with other devices. This is particularly useful for less experienced users, who

can allow their devices to adapt based on the information shared by more experienced users/devices (e.g., less tech-savvy users allowing their thermostats to learn from neighbors' thermostats). These parameters may include air flow, target temperature, and other intelligent thermostat settings.

The *kinship* relationship also enables devices to share errors and repair history, such as an error generated by a boiler and replacement of spare parts or a service record for that particular error. When coupled with other information, *kinships* can also be used to predict a device failure. For example, the usage/load of a furnace together with the probability of the failure of that type of furnace, calculated based on the failure events from other similar devices in *kinships*, enable the owner of the same type of the furnace to receive predicted information about future problems and schedule a maintenance service before the failure occurs.

Device Collaboration for Common Goals

A user's different devices, located in a common space (*co-location relationship*) and/or owned by the same user (*shared ownerships*) can share information among themselves and interact with each other to achieve a common goal. For instance, if a motion detector shares the presence information and a light sensor shares the light level information, then a lamp or a thermostat in the same room can adapt their settings based on the information provided by other devices in relationships. Moreover, any applications that encompass intelligent rules or algorithms applicable to the related device historical data can also take advantage of such information.

Recommendation via Social Navigation

A common way to navigate in an information space in the real world is to leverage the help of other people. The communication with other agents (human or artificial) to navigate in an information space is called social navigation [15, 16, 28]. With the help of new social relationships, it would be possible to navigate and provide recommendations from user to user, based on their friends' networks, same device types (*kinship*), and reputations. This could be done transparently, based on the request from users to their social networks in search of a device of a certain type with certain characteristics. For example, a user posts that s/he would like to purchase a new air conditioner, or the air conditioner posts on behalf of the owner by itself, knowing its lifetime is expiring, describing the characteristics of the current device: 12,000 BTU and top 10 energy efficient air conditioners in *thriendships*. Furthermore, the device itself can initiate the social navigation to inquire about the operational performance of devices in *thriendship*.

5 ONLINE SURVEY: SOCIAL INTERNET OF THINGS FOR SMART HOME SYSTEMS

We conducted an online survey in order to understand and identify users' needs and desired features for the smart home system in SIoT, specifically by asking example scenarios with new social relationship types in Section 4.1. We also collected their subjective opinions and perceived acceptance on the hypothetical SIoT based smart home system, user empowered rule creation, and sharing of their rules with other people. Participants were requested to answer these questions at the end of the survey along with the demographic information.

5.1 Methodology

Recruiting Methods: We recruited 60 participants through Amazon Mechanical Turk (MTurk) [3]. MTurk is an online crowdsourcing labor marketplace where the registered crowd workers perform micro-tasks posted by the requester and get paid based on their submitted results. Qualifications were applied to filter the participants by restricting the survey to the US crowd workers who previously submitted more than 100 tasks and have greater than 95% approval rate. Each participant was paid $1.5 USD for the survey.

Survey Design: The survey questionnaires were designed to have four phases: 1) getting familiar with the smart home and SIoT concepts by watching two videos; 2) describing a user's desire for automated features for smart home systems in general, which implies ownership relationships; 3) describing a user's desire for automated features with consideration of new social relationships as explained in the Section 4.1; 4) answering a set of background questions and a participant's perceived acceptance on the end user programming capability and sharing of their rules and the SIoT concept. All participants were requested to answer the questions in 1), 2) and 3) in text fields except the background questions in 4) which were optional.

In phase 1 of the survey, we asked the participant to watch two smart home promotion videos from LG [24] and Ericsson [17]. The first video introduces a general smart home concept without necessarily connecting to other smart homes. The second video leads the participants to the future with the *Social Internet of Things* paradigm because devices actively communicate and interact with each other as well as with the home owner. The participants were asked to provide one example from the video where two (or more) devices communicate or share information with each other. They were also asked to provide two preferred applications from the videos. We ask these questions to evaluate if the participant had a good understanding of the smart home and SIoT concept through the videos they watched and gained enough context to proceed with next questions.

In phase 2 of the survey, we collected the desired features in a smart home. A picture of a smart home with various devices was presented to the participants to help them come up with two smart home applications by using the devices shown in the picture. This phase focused on only a single smart home, which implies an ownership relationship between the user and devices, if devices are used in their desired application description.

In phase 3 of the survey, we collected the desired features in a smart home given the new social relationships. The new social relationships from the *Socialite* system including *thriendship*, *co-locationship*, *kinship*, and *shared ownership* were explained to the participants with examples of these relationships involving three homes: my home, a friend's home and a stranger's home. For the explanation, a visual description with examples of relationships was used together with textual descriptions. The participants were requested to propose two automated applications leveraging each relationship.

In phase 4 of the survey, the participants were requested to answer demographics, programming experience, social network usage, perceived acceptance of end user programming, sharing of their rules and a SIoT approach/concept similar to the *Socialite* system.

5.2 Demographics

- *Age*: Our participants' ages ranged from 18 to 74. In detail, 40% of the participants were between 25 and 34 years old. 28% were between 35 and 44 years old, 15% were between 45 and 54, 7% were between 54 and 64 years old, 5% of them are each from between 18 and 24 years, and between 65 and 74 years old.
- *Smart home experience*: 10% of the participants have smart home devices at their home.
- *Smart phone experience*: 93% of participants use smart phones.
- *Social networking site experience*: 98% of the participants are currently registered to the social networking sites. 90% of participants have been using the social network sites more than 3 years. Participants use multiple devices to access their social networking sites' accounts. 47% of them spend less than 1 hour per day in the social network site, while 37% spend between 1 and 2 hours, 10% spend between 2 to 3 hours and 7% spend more than 3 hours per day in the social network site.
- *Programming experience*: 55% of the participants do not have programming experience. 15% of participants have less than one year of programming experience, 8% of them have 1 to 3 years of programming experience, 3% of them have 3 to 5 years of experience, and 18% of them have more than 5 years of experience.

5.3 Categorization of SIoT Features

The responses from phase 2 and 3 were 572 smart home features written in English. We qualitatively analyzed the answers through two iterations. During the first iteration of our analysis, the possible feature categories were listed by examining all features collected to abstract them to high-level feature categories. We came up with nine different feature categories. During the second iteration of the analysis, the features from participants' answers were classified into the nine devised categories from the first iteration by labeling each answer with the most relevant category. The labeling was done by two authors of this manuscript. The distribution of the nine categories are illustrated in Figure 3. The description and examples of each category are as follows:

- **Remote access and control**: The features that end users want to monitor and control their connected devices (in ownership relationship) using their smart phone or computer. This feature is same as the feature from the traditional IoT system. Examples from the survey are "Being able to control the thermostat from multiple devices (desktop PC and smart phone, etc.) would be useful." and "Start devices based on my remote input."
- **Device type or capability-based automation**: The features having a combination of device types or capabilities to automate the smart home are classified into this category. Users mention either device types or device capabilities in their answers. The answers of "Automatically lock all doors and windows at the same time." and "I would want the lights to come on when someone rings the doorbell." are examples of this category.
- **Context-based automation**: The definition of the context is often diffused and the perceived context can be different amongst users. The answers that require the combination of multiple devices and/or time and location to determine a condition are classified into this category. If a specified condition can be determined by different rules, an answer is classified into this example. For example, "Turn off TV when not watching." is classified into this category, because determining *when not watching* can be achieved in various ways, for example by detecting user's eye blinking or user's movement. Another example for this category is "When I am going to sleep, turn off room lights". Obviously determining a user's context of *going to sleep* can be done by evaluating a set of device status, time and/or user's location.
- **Preference-based automation**: This category is for the answers that use a person's preferred value for automation rules in a smart home system. An example from the survey is "If my friend's smart phone is in my house, have the heater set to what she normally likes it based on the weather outside." Another example is "Joe's smart phone can ask his smart TV about shows that he likes to watch."
- **Temporal reasoning based automation**: Any answers that require temporal reasoning are classified into this category. Answers may include other automation categories discussed above, but we separate any examples with temporal reasoning because the realization of these features would require different approaches. Examples from the survey are "If the television detects that no one has used the remote *in a while*, send a message to the overhead light to turn itself off." and "If the TV is on, and the motion detects no one is watching TV *after half an hour*, then turn off the TV."
- **Notification**: Answers that include notifying an alert or notifiable state change to the user's smart phone or any

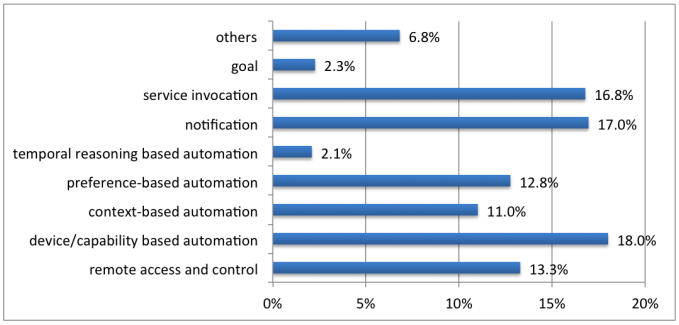

Figure 3: Distribution (%) of the nine feature categories from the user survey analysis

displayable devices such as TV or computer screen. "If my smart phone is in my living room, and my TV is on, then voice mail played on screen." and "The oven in the kitchen finishes cooking and the message is displayed on my smart phone or TV if it is on." are two examples from the survey.

- **Service invocation**: Answers that require further processing of the information beyond the reasoning with the *Socialite* semantic model for devices, users and their relationships. For example, "If I am at home and it is around planned dinner, and nothing has been made, ask me if I'd like to order out." and "If TV is out of order, then ask other TVs in kinship who was used to repair TVs in past and satisfaction rating." are related to this category. First example requires an access to an external service to make an order, and the second example requires a service implementation that can be done by analyzing the repair data stored in the repository.
- **Goal**: Answers that address a high level status of the devices belong to this category. It is expected that this feature category requires a user's input, manufacturer defined value ranges or general consensus on the expectation with a numeric value to validate this feature. A participant responded that "I can check if my Brand X heating system is *working properly based on the usage data* from other Brand X heating systems around me." In this example, *working properly based on the usage data* is ambiguous because it can be reasoned based on the device's energy consumption ranking or the correctness of functions. However, the user's interest is to know a high level status of their devices. This could be achieved with a set of other concrete rules.
- **Others**: Answers that are either not directly related to the features and/or require other research and technology areas beyond this dissertation. For example, security and privacy concerns are important in SIoT but it is not the scope of this paper. Answers that are difficult to understand or are unclear to identify the intended meaning are also belong to this category. For example, answers with device types (e.g., TV) without any application scenarios/descriptions are classified into this category.

5.4 Observation of Relationship Types and Device Life-cycle Relevance

The features from *co-locationship* are mostly used together with *friendship* and *thriendship*. Users want to know preferences of their friends' devices when their friends are users' homes because it helps the user to be more sociable in the offline settings.

The most frequently referred feature for *kinship* includes diagnosis and repair related applications. For this kind of feature, the example scenarios often mention other relationships such as *friendship* and *co-locationship*, which implies that participants like to share with people who they know or can be identifiable because of their proximity. Participants show interests in sharing not only the repair and diagnosis related information but also proper settings and configurations for the devices in *kinship*. We observe that the features listed in the *shared-ownership* relationship address the automated sequence of the actions to achieve a high level task by sharing the device status (e.g., washer is done then start dryer).

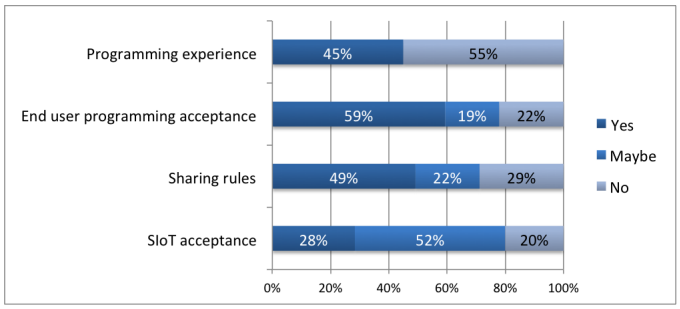

Figure 4: Distribution of programming experience, end user programming acceptance, sharing rules, SIoT acceptance

With the above observation, the *Socialite* reasoning framework (see Section 6) is designed to include the relationship types both in the condition and action expression in the rule description. Moreover, we allow the rules to be able to use union or conjunction of multiple relationships to select the devices that participate in each rule.

5.5 Acceptance of SIoT, End User Programming and Sharing Rules

Figure 4 show the distribution of programming experience, end user programming acceptance, sharing rules acceptance and SIoT acceptance from the participants.

In general, SIoT acceptance is lower than end user programming acceptance and acceptance of sharing of their rules with others. As for SIoT acceptance, the negative answer is low (20%) but the positive answer (28%) is also not high. 52% of participants answered "maybe" for SIoT acceptance. The reasons for the positive acceptance of SIoT concept include that the participants think our approach makes their life easier, secure, safe and save time. Furthermore, participants like the communication and interaction with devices because they can be informed about everything related to home when needed. Amongst the participants who are neutral (answered "maybe"), 48% of them concerned about security and privacy. Therefore, depending on the level of support for security and privacy they can be more positive in using systems in SIoT. Even among the participants with negative answers, 17% of them showed their concern with the security and privacy, while the rest of them are not interested at all in smart home in general.

In a previous study on Facebook privacy [1], the authors reported that privacy was a barrier for people who have not registered to Facebook, while privacy is less concerned for people who are already using Facebook. Also even for users who are using Facebook, their awareness of privacy control on Facebook was misconception.

Many researchers have identified an importance of the security and privacy for Internet of Things. In the SIoT paradigm, the security and privacy would be more concerned as shown in our study. Therefore a further research on what users' perceived privacy risks are and how their behavior would be adapted should be considered as future work.

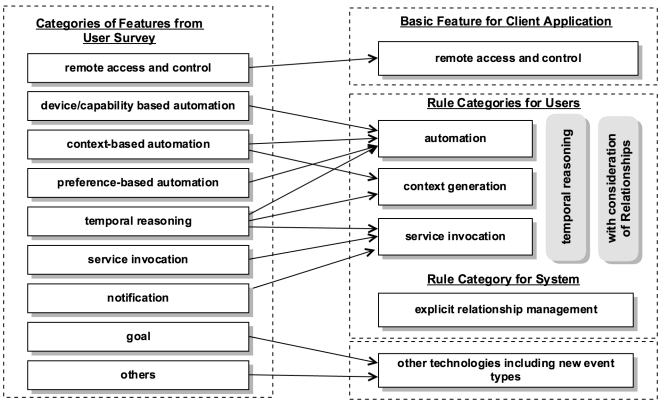

Figure 5: Mapping categorized features from the user survey to rule categories and other functions

The programming experience is not correlated to each of these factors: end user programming acceptance, SIoT acceptance and sharing rules in chi-square tests of independence ($p > 0.05$). Also, neither age nor gender is correlated with these factors in chi-squared test ($p > 0.05$). However, end user programming acceptance, SIoT acceptance and sharing rules are all correlated each other in chi-squared tests of independence ($p < 0.05$). End user programming acceptance is 59% and positive answers on sharing rules is 49%. This results implies that people are open to creating their own rules if an easy to use tool is provided.

6 SOCIALITE REASONING FRAMEWORK

This section discusses the core concept of the reasoning in the *Socialite* system, including the classification of rules and attributes in each rule. The rule categories are driven by the analyzed results from the user survey in Section 5.3 and our system design decision.

6.1 Mapping Feature Categories from Online Survey to Rule Categories

After the feature categorization of all answers from the user survey discussed in Section 5, nine feature categories were mapped into rule categories and other functions (see Figure 5). Two types of user roles are considered in rule creation: end users and system administrators. The rule categories for end users include automation, context generation and service invocation. The rule category for explicit relationship management is assigned for system administrator, because the *Socialite* system aims to provide default rules for explicit relationship management based on our concept of social relationships. In addition, remote access and control feature is assigned as a basic client application feature since it does not require any rules to enable this feature. The rest of two features (goal and others) are mapped to other technologies because the realization of these features requires different technologies (e.g., security and privacy, data mining)

The automation rule category combines device/capability, preference and context based automation where the action in the rule description can be expressed based on the semantic model (See Figure 2). On the other hand, in order to enable context based automation, we introduce the context generation rule category where the user can specify what leads to a user's context. Obviously there exist other technologies to determine a user's context (e.g., applying the Bayesian approach [18]), but this manuscript addresses user-defined context and its reasoning based on the knowledge represented as a rule description. The notification and service invocation features are categorized into the same rule category, because the examples in the notification feature are mostly related to the display type of devices, which are not part of our semantic model and require a service call to follow the manufacturer's notification APIs.

Temporal reasoning and relationships can be used for all the rule categories to express the condition and/or the action in a rule description.

6.2 Classification of Rules

More detail about each rule category are explained with examples as below.

- **Automation based on device capabilities, context, preference and temporal reasoning**

  The *Socialite* reasoning engine supports creating rules to specify automation, that is, the change of a set of device capabilities is triggered based on a specific condition that user defines. Unlike conventional rules used in other smart home systems, the automation rules in the *Socialite* leverage not only a specific device type but also the *device capabilities*.

  Furthermore, the devices in the new social relationships, such as *thriendships* are used in order to express the *condition* and/or *action* in the rule. As for *action* expressions we limit the devices to the ones owned by the user as default. Depending on the access control policies, the devices used in the rules can be selected differently in the future.

  In addition, the *context* defined by the user can be used in expressing *conditions* in the rules. An example of an automation rule, both with device capability and context is "When my living room's temperature is greater than or equal to 75 °F and I am with someone in *friendship*, then set my air-conditioner's target temperature to the temperature that my friend's prefers.". The preference that is accessible via the device semantic model can be used both in the *condition* and *action* expressions to get/set the target value/state in a rule.
- **Context Generation**

  As shown in the analyzed results from the user study (see Section 5), users expect to have automated decisions, not only based on the device status change itself, but also with a higher level situation that can be expressed with the *Socialite* semantic model, information obtained from external services (e.g., weather) and temporal reasoning. The definition of contexts is varied amongst researchers. One of the highly received context definitions is that "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a

user and an application, including the user and applications themselves [13]."

Instead of extracting the context automatically by the system, our solution allows end users to define their own context by using end-user programming. Unlike other smart home systems where the context is a part of the ontology model within the system [37], our system proposes context as a rule generated by a user and sharable with others depending on their privacy settings on the rules. The user-defined context that is inferred by the reasoning engine can be further used by automation rules discussed in the first rule category.

- **Service Invocation** *Socialite* allows *actions* in the rule to be expressed with a service invocation. Services can be provided internally by the *Socialite* system or externally from third party service providers. The services can further use the relationships of the devices and users, a feature that is not yet possible for current practice of the Internet of Things. For example, a service that leverages devices in *kinship* can be used to provide repair solutions by analyzing the repair history of the devices in *kinship*, which had the same error before. Since the *Socialite* semantic model employs the hREST ontology [23] from W3C to represent the REST service interface, any services registered to the *Socialite* system can be expressed in the *action* in the rule. When a rule is fired, a service invocation is called by executing an internal function implemented in the *Socialite* reasoning engine.
- **Explicit Relationship Management** *Socialite* supports specifying a rule to infer explicit relationships. As an illustrated example, let us assume that User 1 owns Device A and User 2 owns Device B. If User 1 and User 2 establish a friendship, or if another device is added to a user's roster of devices, the rule engine may trigger a rule to explicitly create and store new *thriendships*. In addition to the rule for these static relationships, the *Socialite* supports the dynamic aspect of the relationship such as *co-location*. The location of mobile devices (e.g., smart phones) is dynamic in nature; devices may disappear from an area and re-appear in a different environment. The *co-location* relationship rule is triggered based on a device's location. The derived relationships, such as *thriendship* and *kinship* as well as dynamic changes of the *co-locationship* are created by default in the *Socialite* system while *ownership* and *friendships* are controlled by the user. The explicit relationship management rule is configurable to be active or inactive for users who have privacy concerns and do not want the system to create relationships automatically (e.g., automatic creation of co-location relationships with people or devices that a user does not know).

Temporal reasoning in the rule description

The *condition* and/or *action* of all of the rule categories in the *Socialite* system can express temporal aspects of the object with Complex Event Processing (CEP). CEP tracks streams of events (facts in a row) that are inserted into the reasoning engine and detects a specified temporal event in real time. In detail, CEP enables evaluation of whether *event1* happened before or after *event2* or if events happened in a specific time frame. In order to support CEP within a reasoning engine, the Drools engine is configured in a stateful session.

The use of CEP in the *Socialite* aims to identify events in real time over relatively short time periods (less than 30 minutes) either explicitly by specifying the event expiration offset in the rule or implicitly by analyzing the temporal constraints in the rule. Operation related to event detection over long periods of is not addressed in CEP based rule in our system. An example of rules from the user survey is "if my heater's performance is lower than the heater in my friend's home, then notify me with possible solutions". This would be done with other technologies leveraging the harvest data in the device history repository, for example applying abnormal detection algorithms [10]. The rule relevant for this example is rather categorized as a *service invocation* in our rule categories, because it requires to access the device history for long periods which is not stored in the memory in a production rule system, but stored in the device history repository.

Relationship used in the rule description

One novelty of the *Socialite* reasoning concept is that the relationships can be used in all categories of the rules. A rule can specify a specific device by selecting one from the filtered devices and relationship type, or specify a set of devices with relationship type and capabilities.

6.3 Attributes in Rules

The following four attributes are considered in the rule description of the *Socialite* system. These attributes are used for the rule operation and management in the system as well as sharing with others.

- **Sharing attribute:** We expect that the users of the *Socialite* system share their rules to address common goals collaboratively. Towards that end, the *Socialite* reasoning framework supports the sharing attribute. The possible values for the sharing attributes are 1) private, 2) public or 3) constrained to certain relationship types such as friendship and thriendship.
- **Event attribute:** A rule is triggered by an external event that is notified to the *Socialite* reasoning engine in the form of a message from the connected devices, users or relationships. A time event is modeled as a possible value for the event attribute. The state-of-the-art reasoning engine such as Drools [33] provides the time event within the reasoning engine itself, the *Socialite* distinguishes a time event from other events outside the reasoning engine. The time events are further modeled into two perspectives: 1) periodic time event and 2) a time event at a certain point. When a rule from the end user programming is translated into the syntax of the Drools domain specific language, these time perspectives are embedded in a rule description and time events are managed by Drools.
- **Activeness attribute:** A rule is modeled to support activeness attribute so that users can manage their rules to be active or inactive depending on their usage patterns. A

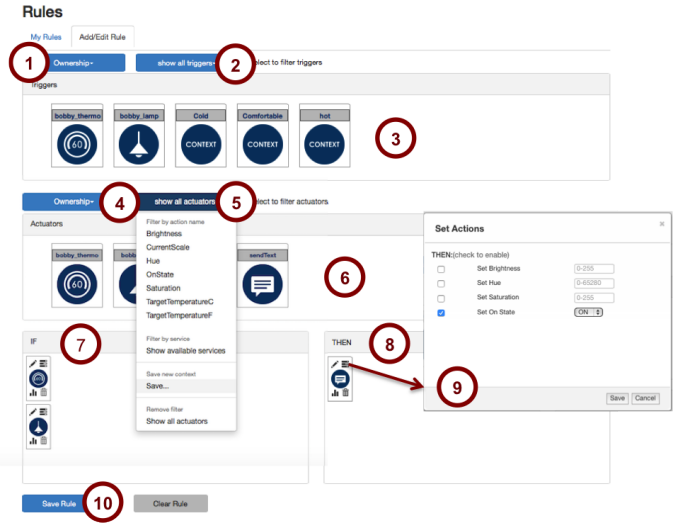

Figure 6: End user programming user interface

rule can be active from now, inactive from now, or active only a certain time period. Operational status based on the activeness can be specified in a higher level rule. For example, a rule can specify that one rule is only active after it meets a certain condition (e.g., the number of execution is greater than a threshold).

- **Goal attribute:** The *Socialite* introduces the goal attribute for further collaboration and information sharing. A set of rules can have a common goal attribute such as energy saving, or securing homes.

6.4 Socialite End User Programming User Interface

The *Socialite* end user programming tool is represented to the user as a trigger-action programming, which is one of the most common formats in the academic literature that matches with users' mental model [12, 14, 35].

Figure 6 represents the main UI view of a rule creation. The devices listed in ① are devices that satisfy the filtering options from ① (*relationship types*) and ② (*device properties* or *contexts*). If the user drags and drops one of the devices or contexts listed in ③ (*Trigger* panel) to ⑦ (*IF* panel), then a pop-up window is shown to allow the user to add a condition (e.g., temperature >80 °F). The user can add as many devices and contexts from ③ as s/he wants to express more conditions. The default operator among condition is conjunction.

In the panel ⑥ (*Actuator* panel), all user's devices with action capabilities and services are shown as a default. The user can select a different relationship type in ④ or filter devices with a certain action capability or services in ⑤. The user can select one of devices or services and drag it to ⑧ (*THEN* panel). The user can set the action value by using the pop-up window (⑨) with a default action pre-selected from ⑤. For the service actuator, the user can select relationship types provided by the service. For example, a "repair solution query service" may ask a user to select kinship or/and thriendship as parameter values used for the implemented service. The user can also create a context instead of device action or service invocation, by clicking the *save context menu* in ⑤.

Once the user finishes the condition and action expression of the rule, then the user clicks *save button* in ⑩, which will call a *Socialite* server API to add the new rule for the current user after providing the rule meta information, such as the rule name and description.

The current implementation of the end user programming application enables the device capability as well as device type based automation, context generation, context based automation, and service invocation with consideration of relationships.

6.5 Production Rules with Semantic Models in Socialite Server

A user-specified rule is created via the end user programming tool. We aim to enable the end-user programming tool developers as well as end users to develop the tool or create rules without detailed knowledge of the domain specific language used in *Drools*. The Drools' rule language is based on declarative programming, and therefore shifting paradigm to a declarative rules style and learning how to write the rules properly and effectively can be time consuming for the client application developers and the rule creators.

In order to address this concern, the *Socialite* server provides rule management interfaces to the end user programming tool so that the client application (which includes end-user programming) can be developed by using REST APIs. The *Socialite* server includes a rule translation mechanism that translates the rule data payload in JSON format to the domain specific language used in the Drools reasoning engine. The JSON payload in the rule creation REST interface is transformed into Java objects. The rule translator uses the Drools APIs to generate the Drools rule objects, which can be put into the reasoning engine.

The *Socialite Server* is an event-driven architecture where asynchronous communication is intrinsic communication mechanism and a basis for our solution to support *scalability*. The reasoning engine that runs rules performs computationally intensive tasks for the evaluate of relevant rules in the engine upon a new event (e.g., device status change). Therefore, reasoning engines are distributed over multiple nodes to evaluates rules for different users in parallel and to accomplish *scalability* by integrating an open source data stream processing solution [34] developed by a social network platform (Twitter).

7 LAB STUDY

We conducted a 24-participant (including 12 non-programmers) user study to evaluate the usability and efficacy of the end user programming module in *Socialite Client*. We had three goals for the study. One was to figure out whether the ideas behind *Socialite Client* are easy to understand and whether the interface is easy to learn and use. The second goal was to compare the performance between programmers and non-programmers thus identify the problems that non-programmers might encounter. The third goal was to investigate whether it is natural for participants to create

Table 2: Rule descriptions used in the user study

Category	Rule
Automation based rules	If the washer is done, send me a text message and set the color of the lamp to be red. If smoke is detected, open the vent and turn on the siren. If motion is detected and TV is on, set the indoor temperature to 70F.
Social relationship based rules	If my friend Jenny's smoke detector detects smoke, send me a text message. If the coffee maker is not working properly, query the diagnosis to my friends who own the same device. If my friend Jenny's siren is triggered, send me a text message.
Context based rules	If I am sleeping, turn off the lamp and the TV. If someone enters my unoccupied house, turn on the siren and send me a text message. If I am watching TV and the room is cold, set the indoor temperature to 75F.

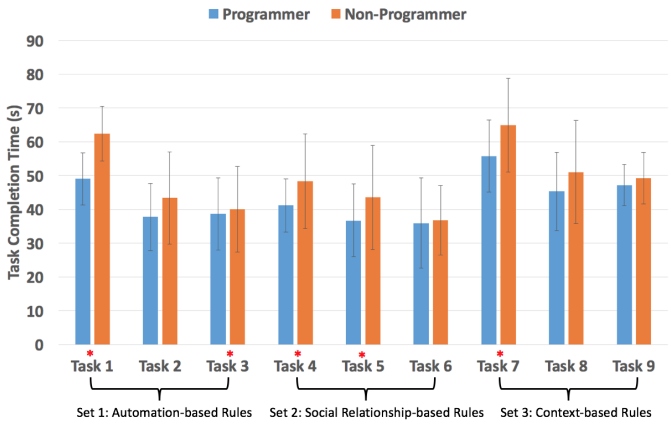

Figure 7: Task completion time of programmers and non-programmers. Significant difference (i.e. p<0.05) between the two groups exists in the tasks with * mark. Error bars show the standard deviations.

social relationship related rules and to explicitly define and use contexts (e.g., while I am sleeping) in rule creation.

7.1 Experimental Design

The study consisted of four parts as follows:

Overview and brainstorming: We first gave participants a brief introduction of SIoT and asked them to watch a video from LG [24] to gain some background knowledge. After that we describes 11 Internet-connected smart devices (temperature sensor, motion sensor, smoke detector, washer, thermostat, door controller, siren, lamp, vent, coffee maker, TV) and 2 envisioned services (send-text-message service, and query-diagnostics service). Participants were asked to write down succinct descriptions of at least 3 envisioned applications (i.e. rules) that they found useful and desirable, assuming they (and also their friends) owned these devices. After that, we used an example rule (if my TV is on, set the temperature to 75F) to teach the participants how to create rules with *Socialite Client*. After the introduction, we let the participants explore the interface freely until they stated explicitly that they were ready to start the follow-up tasks.

Fixed tasks: In this session, participants were asked to create 3 sets of rules to test the general usability of *Socialite Client*. The rules were selected from our previous user survey illustrated in Section 5. Each set contained 3 tasks of roughly equal difficulty. The rules in the first set were straightforward automation based rules (e.g., if the washer is done, send me a text message and set the color of the lamp to be red). Each rule involved 3 devices/services. The rules in the second set were social relationship based rules (e.g., if my friend Jenny's smoke detector detects smoke, send me a text message). The rules in the third set were context based rules (e.g., if I am sleeping, turn off the lamp and the TV). We asked participants to define the contexts (e.g., sleeping) based on their own experience and understanding before using them to create the rules. Table 2 shows the detailed rule descriptions used in the study. In this session we planned to test whether the *Socialite Client* allowed participants to create diffident types of rules in ways natural to them. By comparing user performance (e.g., task completion time) on the three tasks in each task set, we planned to investigate how much participants can improve over time. Thus, the order of tasks was counterbalanced within each task set.

Open-ended tasks and free exploration: In this session, participants were asked to create rules that they wrote down in the previous brainstorming session. Most of the rules overlapped the results from our previous user survey. Participants were encouraged to think aloud and to explore *Socialite Client* as long as they wanted.

Qualitative feedback: After participants completed all the tasks, they were asked to complete a questionnaire and describe their general feeling towards our system. We also conducted semi-structured interviews with participants to solicit their opinions on interface design, their concerns in real world deployment and usage, as well as improvement suggestions they came up with.

7.2 Participants and Apparatus

We recruited 24 participants from two local universities. 12 of them had programming experience (4 females, age from 22 to 41, mean=28.3, SD=7.8, self-reported programming experience was distributed as: <1 year: 1; 1-3 years: 3; 3-5 years: 6; >5 years: 2). The other 12 participants had no programming experience (5 females, age from 19 to 34, mean=25.4, SD=6.3). Each study lasted for around 60 min (up to 90 min maximum), and each participant was given a $10 gift card for the time. An Apple iMac with 1.6GHz dual-core Intel Core i5 CPU, 8GB RAM, 21.5-inch display with a resolution of 1920*1080, running OS X EI Capitan was used for the *Socialite* Client/UI in the user study. The *Socialite Server* was installed in the Amazon Web Service.

7.3 Evaluation Results

Fixed Tasks. All of the participants (i.e. both programmers and non-programmers) were able to create the 9 pre-defined rules through our interface. Participants had no trouble finding the devices, specifying the trigger conditions and the corresponding actions, and creating and saving the rules. Figure 7 shows the task completion time of programmers and non-programmers for the 9 tasks.

The average task completion time of programmers was shorter than that of non-programmers on each of the 9 tasks. T-tests shows that the significant differences exist on task 1 ($t(22)=2.01$, $p=0.01$), task 3 ($t(22)=3.04$, $p=0.05$), task 4 ($t(22)=2.60$, $t=0.02$), task 5 ($t(22)=1.99$, $t=0.05$), and task 7 ($t(22)=3.01$, $t=0.05$). It worth a mention that for the first task of each task set (i.e. task 1, task 4, task 7), the difference on average task completion time between programmers and non-programmers was significant. We attribute this to the large difference in programming experience between the two groups of participants. In other words, the programming experience of programmers might help them create a rule faster than non-programmers, when the type of the rule is new to them. After creating some rules of the same type and complexity, non-programmers can have comparable performance with programmers.

By comparing the task completion time of the first and last task in each rule set, we observed significant improvements. For the automation based rule set, the average completion time of task 3 was significantly shorter than task 1 (51.22s vs. 35.81s, $t(46)=8.99$, $p<0.001$). For the social relationship based rule set, the average completion time of task 6 was significant shorter than task 4 (44.71s vs. 36.33s, $t(46)=4.04$, $p=0.001$). For the context based rule set, the difference between task 7 and task 9 was also significant (60.33s vs. 48.19s, $t(46)=6.67$, $p<0.001$). This implies that the *Socialite Client* had a low learning curve, thus participants were able to create rules of roughly equal complexity in shorter amounts of time.

Free exploration. The participants came up with a total of 79 envisioned rules. Most of the rules overlapped the results from our previous study discussed in Section 5. Participants were asked to implement their open-ended rules as close to their written specifications as possible, and they successfully implemented every one of the rules they described. Participants found that the rules they desired are "similar with the tasks" (P5) that they were asked to complete and thus it was "not hard at all" (P21) to create their own rules after some practice. On average participants spent around 7 minutes in this session creating their own rules and exploring the interface.

There were 6 participants that totally ignored the feature of filtering devices by capabilities. For example, in order to create the action "set temperature to 75F", users can either first select the "temperature" capability and then choose the device (e.g., thermostat) on which they can set the value, or select the device directly without the filtering step. The reason why they did not use this feature was that they "already know the capabilities of each device quite well" (P3). They mentioned that this feature could be important when "there are a lot of devices" (P10), and the devices have "complex capabilities" (P19). Beside these 6 participants, other participants adopted a mixed approach among capability selection and device selection.

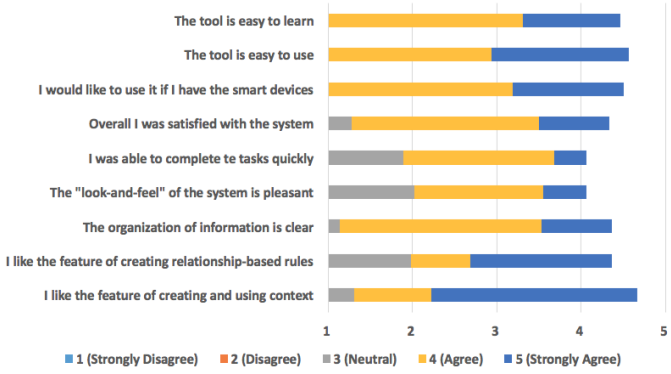

Figure 8: Subjective ratings on a 5-point Likert scale (1 = strongly disagree, 5 = strongly agree). Length of a bar denotes the average rating. Color encoding shows the distribution of participants' responses.

Subjective Feedback. Overall, participants reported positive experiences with *Socialite end user programming interface*. Participants felt that the system was easy to learn and use, and provided ways that were natural for them to create rules (Figure 8). Here we listed some sample responses from our participants:

- "I'd like to use it in my house and working place, because it can definitely save my time and make life more convent and safe" (P3)
- "It's easy to learn and well visualized. Meanwhile it feels like very user-oriented and convenient" (P11)
- "We do not need to write program for creating those rules. It's good for non-programmers, it's also good for programmers like us" (P14)

Participants also provided some improvement suggestions based on the current design. One common suggestion was to "combine triggers and actuators" (P8, P19, P20) to save screen space. Since we already provided instantaneous feedback on where to drop the device once the user began to drag it, they believe "there is no need to separate them into two groups" (P20). Participants also mentioned their concerns on privacy and security of enabling social relationship based rules. Some participants were "not sure about the authentication mechanism" (P2), and were afraid that "someone can fully control my house" (P9). But all of the participants expressed their willingness of using the system if they can fully control the authentication of their own devices. While the main focus of this paper is to demonstrate the concept and feasibility of enabling end users to create social relationship based rules, it is out of the scope to address the security and privacy issues involved. However, it is one important future work direction.

8 DISCUSSION AND CONCLUSION

We presented an end user programming tool for the new paradigm of SIoT, where people and devices can use in smart homes to discover and share information toward an effective collaboration in an autonomous and personalized way. Our SIoT instantiation is

supported by newly defined social people-device and device-device relationships, which serve as a foundation for a new framework for the SIoT.

Depending on the user's needs in different phases of the device life-cycle, different relationship types can be properly utilized. For example, we defined a new social relationship between devices that can be used to discover and share repair history and solutions.

Furthermore, the end user programming tool we introduced makes it easy for end users to create their own rules and to share them with others. The basic information of the user, device and its diagnosis, location, relationship and service represented as semantic models is used when end users define their own rules. In a user study, including 12 participants with no programming experience, participants were able to use *Socialite* to create automation based rules, rules that involved social relationship, and new rules participants desired, after spending a small practice.

The user survey in this manuscript uncovers that the user's acceptance of the SIoT is highly related to how the system supports security and privacy, which is also supported by a survey from Pew Research Center [26]. Somewhat contradictorily, participants have demonstrated that they want to offer and get help with their devices and rules. The introduction of new relationships will open the discussion of balancing sharing/openness benefits and privacy risks from sharing, given the gap between their perceived privacy risks before and after joining the system. Note that privacy in SIoT will take different forms and will depend on device types (e.g., home appliances, health monitoring devices, security devices), different phases in the device life-cycle (e.g., operation, maintenance phases), data processing phases (communicating, processing, storing and sharing data), and interaction with devices and other users in different social relationships. Privacy enhancing solutions for the collaboration framework will be able to bootstrap a wide adoption of the new paradigm of the SIoT and ultimately help to gain a new form of social capital obtained from new social networks with people and connected devices.

REFERENCES

[1] Alessandro Acquisti and Ralph Gross. 2006. Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook. In *Proceedings of the 6th International Conference on Privacy Enhancing Technologies (PET'06)*. Springer-Verlag, Berlin, Heidelberg, 36–58.

[2] Paul S Adler and Seok-Woo Kwon. 2002. Social capital: Prospects for a new concept. *Academy of management review* 27, 1 (2002), 17–40.

[3] Amazon.com Inc. *Amazon Mechanical Turk*. Amazon.com Inc. https://www.mturk.com

[4] Atooma 2017. . Atooma. https://www.atooma.com

[5] Luigi Atzori, Davide Carboni, and Antonio Iera. 2014. Smart things in the social loop: Paradigms, technologies, and potentials. *Ad Hoc Networks* 18 (2014), 121–132.

[6] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2011. SIoT: Giving a social structure to the internet of things. *Communications Letters, IEEE* 15, 11 (2011), 1193–1195.

[7] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. 2012. The Social Internet of Things (SIOT)–When social networks meet the internet of things: Concept, architecture and network characterization. *Computer Networks* 56, 16 (2012), 3594–3608.

[8] Matthew Ball and Vic Callaghan. 2012. *Managing Control, Convenience and Autonomy-A Study of Agent Autonomy in Intelligent Environments*.

[9] M Baqer. 2010. Enabling collaboration and coordination of wireless sensor networks via social networks. In *Distributed Computing in Sensor Systems Workshops (DCOSSW), 2010 6th IEEE International Conference on*. IEEE, 1–2.

[10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41 (2009), 15.

[11] James S Coleman. 1988. Social capital in the creation of human capital. *American journal of sociology* (1988), S95–S120.

[12] Luigi De Russis and Fulvio Corno. 2015. HomeRules: A Tangible End-User Programming Interface for Smart Homes. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2109–2114.

[13] Anind K Dey. 2001. Understanding and using context. *Personal and ubiquitous computing* 5 (2001), 4–7.

[14] Anind K Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive prototyping of context-aware applications. In *Pervasive Computing*. Springer, 254–271.

[15] Andreas Dieberger. 1997. Supporting social navigation on the World Wide Web. *International Journal of Human-Computer Studies* 46, 6 (1997), 805–825.

[16] Paul Dourish and Matthew Chalmers. 1994. Running out of space: Models of information navigation. In *Short paper presented at HCI*, Vol. 94. 23–26.

[17] Ericsson 2011. *The Social Web of Things*. Ericsson. https://www.youtube.com/watch?v=i5AuzQXBsG4

[18] Tao Gu, Hung Keng Pung, Da Qing Zhang, Hung Keng Pung, and Da Qing Zhang. 2004. *A bayesian approach for dealing with uncertain contexts*. na.

[19] Dominique Guinard, Mathias Fischer, and Vlad Trifa. 2010. Sharing using social networks in a composable web of things. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, 702–707.

[20] IFTTT Inc 2017. . IFTTT Inc. https://ifttt.com

[21] Aman Kansal, Suman Nath, Jie Liu, and Feng Zhao. 2007. Senseweb: An infrastructure for shared sensing. *IEEE multimedia* 4 (2007), 8–13.

[22] Ji Eun Kim, Adriano Maron, and Daniel Mosse. 2015. Socialite: A Flexible Framework for Social Internet of Things. In *Mobile Data Management (MDM), 2015 16th IEEE International Conference on*, Vol. 1. IEEE, 94–103.

[23] Jonathon Kopecky, Karthik Gomadam, and Tomas Vitvar. 2008. hrests: An html microformat for describing restful web services. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, Vol. 1. IEEE, 619–625.

[24] LG Global. *CES LG - Smart Home*. LG Global. https://www.youtube.com/watch?v=-AsuUdi1BiY

[25] Nan Lin. 1999. Building a network theory of social capital. *Connections* 22, 1 (1999), 28–51.

[26] Mary Madden. 2014. *Public Perceptions of Privacy and Security in the Post-Snowden Era*. http://www.pewinternet.org/2014/11/12/public-privacy-perceptions/.

[27] Niko Mäkitalo, Jari Pääkkö, Mikko Raatikainen, Varvana Myllärniemi, Timo Aaltonen, Tapani Leppänen, Tomi Männistö, and Tommi Mikkonen. 2012. Social devices: collaborative co-located interactions in a mobile cloud. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 10.

[28] Alan J Munro, Kristina Höök, and David Benyon. 2012. *Social navigation of information space*. Springer Science & Business Media.

[29] Mark W Newman. 2006. Now we're cooking: Recipes for end-user service composition in the digital home. (2006).

[30] Huansheng Ning and Ziou Wang. 2011. Future Internet of things architecture: like mankind neural system or social organization framework? *Communications Letters, IEEE* 15, 4 (2011), 461–463.

[31] Antonio M Ortiz, Dina Hussein, Soochang Park, Son N Han, and Noel Crespi. 2014. The cluster between internet of things and social networks: Review and research challenges. *Internet of Things Journal, IEEE* 1, 3 (2014), 206–215.

[32] Antonio Pintus, Davide Carboni, and Andrea Piras. 2012. Paraimpu: a platform for a social web of things. In *Proceedings of the 21st international conference companion on World Wide Web*. ACM, 401–404.

[33] Mark Proctor. 2011. Drools: a rule engine for complex event processing. In *International Symposium on Applications of Graph Transformations with Industrial Relevance*. Springer, 2–2.

[34] Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M Patel, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, and others. 2014. Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 147–156.

[35] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. 2014. Practical Trigger-action Programming in the Smart Home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 803–812.

[36] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3227–3231.

[37] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. 2004. Ontology based context modeling and reasoning using OWL. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*. Ieee, 18–22.

[38] WigWag Inc 2017. . WigWag Inc. https://wigwag.com