Semantic Integration of Sensor Measurements and Human Self-observations for Physical-Cyber-Social Computing

Artem Katasonov, Jarkko Leino and Timo Tuomisto

VTT Technical Research Centre of Finland
Tampere, Finland
e-mail: artem.katasonov@vtt.fi, jarkko.leino@vtt.fi, timo.tuomisto@vtt.fi

***Abstract*—To deliver on their promises, Physical-Cyber-Social systems must semantically integrate a wide range of heterogeneous multimodal observations, including physical sensor measurements, human self-observations, social observations, and demographic observations. In this paper, we address the problem of collecting and combining sensor measurements and self-observations. We describe an architecture, main features of which are a triple-space-based approach to data integration and a novel approach utilizing instant messaging for eliciting self-observations. A specific application domain considered is health-related monitoring of elderly persons at their homes.**

Keywords – physical-cyber-social; smart home; data integration; assisted living

I. Introduction

Amit Sheth has recently popularized the concept of *Physical-Cyber-Social (PCS) computing* [1] as an emerging paradigm supported by the expanding Internet of Things (an improved ability to observe the physical world), cyberspace (an improved ability to access a massive repository of background knowledge on the Web), and social media (improved access to social knowledge). [1] claimed that PCS will be able to address in a holistic manner questions that neither human intelligence nor present computing systems can answer. In healthcare, for example, this could mean a highly personalised interpretation of a blood pressure reading and personalised suggestions of corrective actions. More generally, the combination of cyber-physical and social data can help us to understand events and changes in our surrounding environments better, monitor and control buildings, homes and city infrastructures, provide better healthcare and elderly care services among many other applications [2].

[1] states that in order to achieve its goals, PCS must access and semantically integrate a wide range of heterogeneous multimodal observations, including physical sensor measurements (such as blood pressure or heart rate), self-observations (subjective states and sensations), social observations (from a network including family, friends, and colleagues), and demographic observations (aggregated characteristics of the population with similar attributes or lineage). In addition to such integration of observations performed by PCS horizontal operators [1], there is a need for PCS vertical operators to progressively lift the integrated observations along the Data-Information-Knowledge-Wisdom dimension. Horizontal operators deal with variety and veracity of data, while vertical operators deal with its volume and velocity.

In this paper, we introduce a software architecture aiming to provide one required horizontal operator for PCS systems that is the collection and semantic integration of physical sensor measurements and human-produced observations, in particular, self-observations. The main features of this architecture include a triple-space-based approach to open interconnected systems [3], use of W3C's Semantic Sensor Network (SSN) ontology [4], an automation framework for triple-space management, a novel approach utilizing instant messaging for eliciting human-produced observations and integrating them with the rest of data, and optional integration of the speech interface within the latter. Collection and processing of human observations has been a rapidly growing research area [5] but focused on the analysis of unsolicited observation streams, e.g., from Twitter. To the best of our knowledge, there exists no streamlined approach to collecting solicited human observations and integrating them with physical sensor measurements.

The rest of the paper is structured as follows. Section II describes one motivating scenario for this work, namely health-related monitoring of an elderly person at home. Section III specifies the main components of our data integration architecture, with a brief discussion of related security aspects. Section IV follows with a description of our approach to collecting self-observations, including how we integrate a speech-based interface as a part of it. Finally, Section V concludes the paper.

II. Motivating Case: USEFIL

This work is performed in and motivated by the application domain of the EU FP7 project *Unobtrusive Smart Environments for Independent Living (USEFIL)* [6], [7].

The life expectancy in the EU and in other developed countries is continuously increasing and the proportion of elderly citizens in the population is growing. The elderly are often living alone, approximately one of every three non-institutionalized older adults [8], and often cannot afford private carers. Prolonging their ability to remain independently in their homes may yield economic and emotional benefits at the societal and personal levels. On the other hand, elderly are prone to decline in physical abilities, chronic illnesses, cognitive decline, and depression. Therefore, responsibly postponing their move into a nursing home requires maintaining a continuous confidence in their ability of independent and safe living.

The most common reasons elderly are admitted into nursing homes are caregiver burden and the elderly person's inability to perform Activities of Daily Living (ADLs) such as moving around, dressing, or bathing [9]. The elderly are also especially affected by loneliness and depression brought on by the Empty Nest Syndrome or neglect, either intentional or not [9]. In line with this, the main objective of the USEFIL project is to provide low-cost ICT solutions for monitoring physical health, cognitive health and emotional status of elderly (65+ years old) with age related disabilities at their homes – to assess their ability of independent living and to detect deteriorations when they occur. In addition, USEFIL aims at facilitating elderly access to telecare, as well as supporting their social interactions to fight loneliness.

The setup of the USEFIL system allows for various physical sensors. A central role is played by imaging devices: a video camera placed behind a mirror and a depth sensor such as Microsoft Kinect. The project develops algorithms to extract from those data various ADL parameters, such as walking balance and ability of transfer (e.g., raising from a chair), health signs, such as heart rate and pupils equality (relevant to stroke patients), as well as emotional states. In addition, USEFIL features a wrist wearable unit (Android watch-phone) with custom algorithms for processing accelerometer data to recognize and monitor daily activities.

As end-user interfaces, USEFIL employs a Smart TV and a tablet computer. Both provide a user interface for data access (health trends, medication schedule and reminders, etc.) as well as communication channels, both to healthcare and to family/friends. The wrist unit also provides a limited user interface allowing receiving messages as well as including a software "panic" button.

In addition to physical sensors for unobtrusive monitoring, USEFIL realized a need for collecting self-observations, where the monitored elderly person is posed one or more questions to answer using either TV or tablet interface. Self-observations complement the physical sensor observations and are needed, in particular, for accessing the emotional status of an elderly person and detecting depression.

The USEFIL system includes data fusion and Decision Support System components which are responsible for abstracting the data, thus can be seen as PCS vertical operators (see Section I). In the following sections, we describe the architecture responsible for the collection and integration of data, which can be seen as a PCS horizontal operator.

III. Data Integration Architecture

A. Semantic Integration

The proposed here solution to heterogeneous data collection and integration is based on a Triple Space [3] as the approach to implementing an open interconnected system. A triple space is a special case of a tuple space. A tuple space is an implementation of an associative memory (also known as blackboard or distributed shared memory) for parallel or distributed computing. A repository of tuples (ordered lists of data elements) is provided that can be accessed concurrently; producers post their data as tuples in the space and consumers retrieve data from the space using queries or a subscription mechanism. In result, most of the direct communication between system components is substituted with posting to and reading from the tuple space. Such an approach separates the data themselves from such questions as data availability (where to find it?) and transmission (when and where to send?), thus greatly simplifying distributed application development. This paradigm has become quite popular in the Internet of Things field with many storage and integration Cloud services, such as Xively (formerly Cosm and before that Pachube), being such tuple spaces.

Semantic technologies based on machine-interpretable representation formalism have shown promise for describing objects, sharing and integrating information, and inferring new knowledge [10]. Therefore, [10] states that utilisation of semantic technologies is important for interoperability, data integration, data abstraction and access, resource search and discovery as well as reasoning and interpretation on the Internet of Things. Also, [1] argued for semantics in a wider context of PCS systems (see Section I).

Merging the tuple space paradigm with semantics, the result is a triple space where all the tuples are semantic triples {subject, predicate, object}. Some proprietary triple-space-based approaches were developed, including generic, such as [3], and specifically with resource-limited devices in mind, such as Smart-M3 [11]. However, with the present state of the technology, a triple space can also be set up without any proprietary software – via deploying an off-the-shelf RDF data server and posting and reading data using the standard SPARQL language and protocol.

In our solution, we followed the latter approach of relying on standards and reliable data storage products. In particular, we utilise an OpenRDF Sesame database deployed on an Apache Tomcat HTTP server. Data producers interact with various sensors using the special protocols that those sensors support and publish the observations to a Sesame repository as RDF triple-sets (see below). Data consumers never interact with data producers directly and only look for needed observations to appear in the database. Data prosumers, again, do not interact with any data producers or consumers directly, but read data from the database and output their results back into the database. All of these communications occur over the interface offered by the database which is SPARQL 1.1 Protocol over HTTP with SPARQL 1.1 Query or SPARQL 1.1 Update payloads. To facilitate data producer/consumer programming, we developed, however, software libraries (Java and C) hiding the HTTP and SPARQL operations and offering a simple to use programmatic API.

One drawback of using an off-the-shelf RDF database instead of a proprietary triple space solution is that subscriptions with push-notifications are not available to data consumers – a mechanism typically included into proprietary systems [3], [11]. A subscription mechanism offers two main advantages: (1) more convenient client programming and (2) avoiding the database and the network load with frequent repeated queries, or introducing a delay to reacting to new data. To address the latter and more important issue, our solution includes a *Change Notifier* service, which is a simple Web service (Java servlet) deployed on the same Tomcat instance as the Sesame database. It operates based on the 'long-polling' model. This means that, after receiving a request, it waits for up to the specified timeout before responding with a response indicating that that no change occurred in the database. If any change occurs in the database during the waiting time, the service will immediately return and provide the timestamp of the change. One may use also 'since' request parameter with a timestamp. This simple service allows avoiding unnecessary repetition of queries, as well as executing a query immediately after some new data become available, providing close to real time reaction as often required in Internet of Things environments.

The conceptual data model of our solution is principally based on the W3C's SSN Ontology [4]. The SSN ontology is a domain-independent ontology that describes sensors and observations by merging sensor-focused, observation-focused and system-focused views. SSN is based itself on the DUL (DOLCE Ultra Light), with DOLCE standing for Descriptive Ontology for Linguistic and Cognitive Engineering. DUL is an upper ontology that defines only 5 classes at the top level of the concepts hierarchy, Object, Quality, Event, Abstract, and InformationRealization, as well as a larger set of properties for describing possible relationships between instances of these classes and their subclasses. Most of the concepts defined in SSN are then subclasses, sub-properties, or other derivatives of DUL concepts.

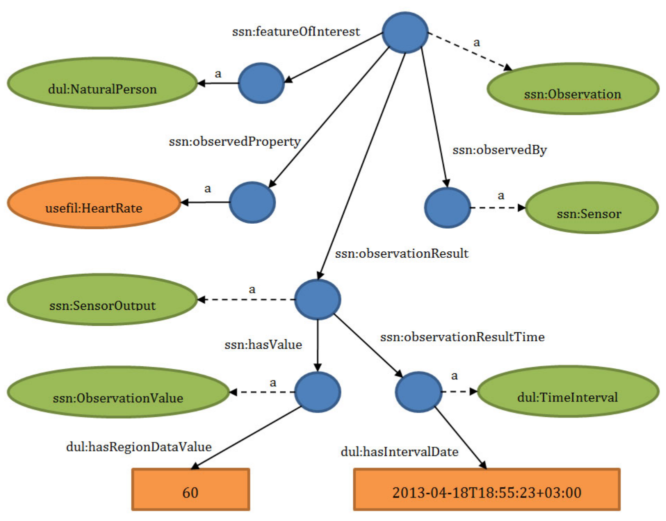


Fig. 1. An example of the description of a sensor observation.

Following SSN, we conceptually describe a sensor observation as depicted in Figure 1. Practically, we define and use, however, a more compact representation which reduces the number of triples and introduces only one anonymous node per measurement instead of seven in Figure 1 (blue circles). This is done without losing semantics via use of the OWL2 property-chain mechanism. In terms of Figure 1, every important path from the root to a leaf is substituted with a single custom property. For example, *usefil:observationResult* is defined as owl:propertyChainAxiom (ssn:observationResult ssn:hasValue dul:hasRegionDataValue).

With respect to required domain-specific concepts, such as types of observations (usefil:HeartRate in Figure 1, not covered by SSN or DUL), our approach is to either use concepts from established domain ontologies or to define custom concepts as subclasses of those. In the case of the USEFIL system, we utilised such ontologies as International Classification of Functioning, Disability and Health (ICF), International Classification of Diseases (ICD-10), SNOMED Clinical Terms, and Clinical Measurement Ontology (CMO).

B. Tasker Framework

To support automated management of a triple space, we have defined and implemented a software tool we refer to as the *Tasker framework*. Tasker allows timed and repeated execution of various SPARQL tasks on an RDF database, as well as execution of external Java code. It can be used as a library or a stand-alone application, in both cases controlled fully via its configuration file. Such a configuration file is encoded using Turtle RDF and contains definitions of one or more tasks. An example of a task follows.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix t: <http://www.vtt.fi/usefil/tasker#> .
[a t:Task]
    t:prefixes "PREFIX usefil: <http://usefil.eu/ontology#>" ;
    t:where "?event usefil:observationResultTimeMillis ?time.
      FILTER (?time < %NOW% - 3600000). ?event ?p ?o" ;
    t:delete "?event ?p ?o" ;
    t:construct "?event ?p ?o" ;
    t:execute "fi.vtt.usefil.tasker.executable.Backup" ;
    t:start "12:00" ;
    t:repeat "600000" ;
    rdfs:comment "Remove old events" .
```

Each task is a *t:Task* instance and can have the following properties (all of which are optional):

- *rdfs:comment* Name of the task.
- *t:enabled* If false, the task is inactive.
- *t:prefixes* Task-specific SPARQL prefixes (other than rdf:, rdfs:, owl:).
- *t:where* Corresponds to WHERE clause of SPARQL. Used in a SELECT query and/or INSERT, DELETE, CONSTRUCT queries.
- *t:insert* Corresponds to INSERT clause of SPARQL.
- *t:delete* Corresponds to DELETE clause of SPARQL.
- *t:construct* Corresponds to CONSTRUCT clause of SPARQL.
- *t:execute* A Java class name, optionally followed by command line arguments, to be dynamically loaded and

executed. This class has to implement a predefined interface through which Tasker will feed it the results of execution of SELECT and/or CONSTRUCT queries.
- *t:start* Start time (first run) for the task. Can be an ISO 8601 time (e.g., 2013-11-10T12:00:00+0200) or a number in milliseconds which is interpreted as delay from the initialization time. Also partial forms of ISO 8601 time are accepted, like 12:00 above (which assumes current date and local time zone).
- *t:repeat* Repetition interval for the task, in milliseconds. Additional values are 0 meaning execution only once and -1 meaning execution every time a change occurs in the database (see Section III-A about Change Notifier).

The above example task thus removes from the database all the observations older than one hour (3600000 ms), while backing them up as RDF into a file (the *Backup* executable does that for the CONSTRUCT query result). This task is executed every 10 minutes (600000 ms) starting at noon.

Using different combinations of the task properties, Tasker can achieve a variety of goals, including data management (as in the example), rule-based reasoning / data fusion, as well context-dependent execution of application code.

C. Secutity Aspects

Along with the design of the above data integration architecture, an extensive analysis was performed of its security properties. Threats and vulnerabilities related to different security domains [12] were listed and addressed via specific measures. This analysis is outside the scope of this paper; below we only list the most prominent aspects.

The database server has to enforce secure communication, i.e., HTTPS, even if only used within a wireless network protected using WLAN security. This is to authenticate the server to data producers/consumers to prevent client spoofing by a fake server, as well as to prevent wireless eavesdropping among connected devices. The server also has to require the clients to authenticate for both data access and publication. This protects data from unauthorised access and prevents database spoofing, where a malicious software process publishes fake observations. Specifically, we register each data producer separately (own username and password) and insert any triples it submits into a separate sub-graph of the database, facilitating provenance tracking and non-repudiation. Data producers have to be designed to only push measurements to the database and not to implement any own query interfaces that would require a separate protection, which may be difficult. Finally, all the clients have to avoid storing any private data into temporary local files, i.e., should rely only on the database for persistence.

IV. Collecting Human Observations

A. Management of a Machine-Human Dialog

In the context of the general architecture described in Section III, collecting human self-observations is an issue of design of a specific type of a data producer.

Systematic collection and processing of human observations in general has been a rapidly growing topic in the research community. In particular, may studies address citizen sensor networks [5], which refer to interconnected networks of people who actively observe, report, collect, analyze, and disseminate information via text, audio, or video messages. These approaches focus on the analysis of readily-available *unsolicited* observation streams, in particular from microblogging systems such as Twitter. In contrast, we are interested in *solicited* observations, such as self-observations of a personal state. For solicited observation, Web-based online questionnaire forms remain a dominant tool, while there seems to be no streamlined approach to integration of answers from such online forms with physical sensor measurements.

In our solution, a questionnaire is defined as a state-machine and encoded in an XML document. In addition to messages, questions and answer options, this document includes all the information needed for the interpretation of answers in terms of our conceptual data model as well as accessing data from the triple space to personalise the questions. A state-machine representation allows two ways of administering a questionnaire. For longer questionnaires, we automatically transform them into Web forms. For shorter questioning sessions, we are also able to administer them question-by-question via an instant messaging protocol. For the latter case, we implemented both text-based and speech-based interfaces. For the text-based option, we developed a custom Android chat app that renders the answer options as buttons in the chat window (Figure 3). The use of a standard XMPP chat client is also possible, except for that the answers have to be typed then.

Figure 2 provides an example of a questionnaire definition that encodes the first question from Beck Depression Inventory (BDI) and is accompanied by a preamble and a closing message.

A questionnaire consists of *d:Message* and *d:Question* elements, the order of which is defined via their *d:state* and *d:transition* attributes. Upon a message, transition to the new state is done immediately. Upon a question, the transition is done only after a valid answer is received. The starting state is called ”start” and the final one ”end”. While a message only has *d:Text* property (the message itself), a question also has to have: *usefil:observedProperty*, which is the type of observation collected with this question, and a set of possible answers to the question. Each *d:Answer* has:
- *d:Text*. Text to display.
- *d:Value*. Identifier of the answer option, e.g., ”0”. If the user interface does not show answer options as list or buttons, but requires typing or speaking, values are used as accepted entries.
- *d:Shortcut* (optional). Another accepted entry corresponding to the answer option, e.g., ”no”.
- *usefil:observationResult*. Value for the evaluated usefil:observedProperty if this answer is selected.
- *d:Message* (optional). A message to deliver back to the user as a feedback to the answer.

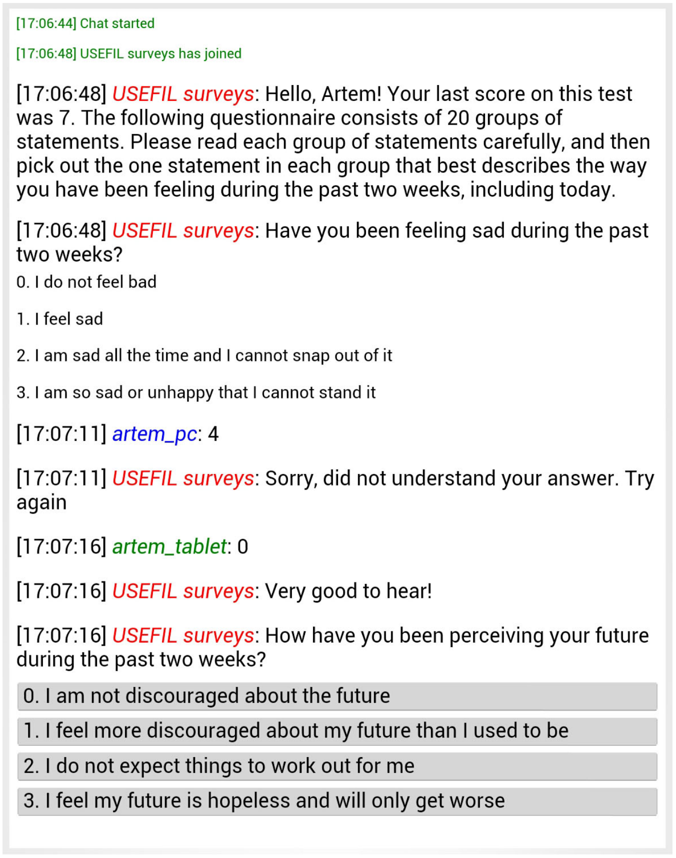
```
<d:Questionnaire xmlns:usefil="http://usefil.eu/ontology#"
    xmlns:d="http://usefil.eu/dialog#"
    xmlns:db="http://usefil.eu/database#" id="BDI" >

  <d:Message d:state="start" d:transition="1">
    <db:Query db:type="sparql"><![CDATA[
      PREFIX usefil: <http://usefil.eu/ontology#>
      SELECT * WHERE {
        [a usefil:User] usefil:hasName ?name .
        [a usefil:Measurement]
          usefil:observedProperty usefil:BDI;
          usefil:observationResult ?beck_score }
    ]]></db:Query>
    <d:Text>
      Hello, %name%! Your last score on this test was
      %beck_score%. The following questionnaire consists of ...
    </d:Text>
  </d:Message>

  <d:Question d:state="1" d:transition="2">
    <d:Text>Have you been feeling sad
        during the past two weeks?</d:Text>
    <usefil:observedProperty>usefil:FeelingSad
        </usefil:observedProperty>
    <d:Answer d:transition="3">
      <d:Shortcut>no</d:Shortcut>
      <d:Text>0. I do not feel bad</d:Text>
      <d:Value>0</d:Value>
      <usefil:observationResult>0.0</usefil:observationResult>
      <d:Message>
        <d:Text>Very good to hear!</d:Text>
      </d:Message>
    </d:Answer>
    <d:Answer> ... <d:Answer>
    ...
  </d:Question>

  <d:Question d:state="2"> ... </d:Question>
  ...
  <d:Message d:state="21" d:transition="end">
    <d:Text>Thank you for your answers!</d:Text>
  </d:Message>

</d:Questionnaire>
```

Fig. 2. The questionnaire definition format.

- *d:transition* attribute (optional). Overrides that on the question, e.g., answering "no" to the example question will lead to skipping the next question.

Any Message or Question can also have an optional *db:Query* property, giving a query that has to be executed on the triple space prior to delivery of the message/question. The values retrieved for the query variables will substitute corresponding placeholders in *d:Text* content.

Our implementation of the instant messaging approach is based on the XMPP messaging protocol. A user can have a number of user interface devices connected simultaneously; in USEFIL, this includes Smart TV, tablet, wrist-wearable unit, and a speech interface (on a PC). A question is received and rendered on all currently connected devices and can be answered from any of them. In USEFIL, we plan to use


[17:06:44] Chat started

[17:06:48] USEFIL surveys has joined

[17:06:48] USEFIL surveys: Hello, Artem! Your last score on this test was 7. The following questionnaire consists of 20 groups of statements. Please read each group of statements carefully, and then pick out the one statement in each group that best describes the way you have been feeling during the past two weeks, including today.

[17:06:48] USEFIL surveys: Have you been feeling sad during the past two weeks?
0. I do not feel bad

1. I feel sad

2. I am sad all the time and I cannot snap out of it

3. I am so sad or unhappy that I cannot stand it

[17:07:11] artem_pc: 4

[17:07:11] USEFIL surveys: Sorry, did not understand your answer. Try again

[17:07:16] artem_tablet: 0

[17:07:16] USEFIL surveys: Very good to hear!

[17:07:16] USEFIL surveys: How have you been perceiving your future during the past two weeks?

0. I am not discouraged about the future

1. I feel more discouraged about my future than I used to be

2. I do not expect things to work out for me

3. I feel my future is hopeless and will only get worse

Fig. 3. An example of a questionnaire chat session.

the instant messaging approach not only for soliciting health-related self-observations, but also in a number of other tasks. This includes validation of systems interpretations ("do you need to call somebody for help?", "does your wrist unit have to be charged?") or to collect feedback ("was it easy to use the application?").

B. Speech Interface

A number of studies [9], [13] argued that, especially in the case of an ICT system assisting elderly persons, the use of speech-based interfaces is beneficial and may increase the end user acceptability of the system.

The text-to-speech transformation needed for a message/question delivery is a simpler problem supported by a number of reliable tools. Speech recognition, on the other hand, is shown to be a particularly hard problem to be solved reliably for most environments. Our approach to collecting self-observations, however, presents a very restricted case of the general speech recognition problem. First, in our case, only the machine can initiate the dialog by posing a question. Therefore, we do not face a hard problem of interpreting a free-form user command, which is a case for most speech-based systems. Second, the vocabulary that the user is expected to use when answering a question is predefined and very limited. In our XML format, the answering vocabulary for a question is the union of *d:Value* and *d:Shortcut* properties of all defined answer options. For the example question is Section IV-A, such vocabulary consists of eight words '0', '1', '2', '3', 'no', 'yes', 'always', and 'unbearably'. Speech recognition in the case of such a very limited vocabulary is shown to be a much simpler and manageable problem.

We implemented a speech interface as an XMPP client, which thus can be used along with text-based XMPP clients. Any message/question received from the system is translated to speech using Mary TTS system. User's answer is recognised using Simons Listens system and, if it is deemed to belong to the acceptable vocabulary, translated into XMPP and sent back to the system. Simon Listens supports dynamically restricting the expected vocabulary, providing a sufficient performance level in an indoor environment.

V. CONCLUSIONS

In this paper, we introduced a software architecture aiming to provide one required horizontal operator for Physical-Cyber-Social systems that is the collection and semantic integration of physical sensor measurements and human-produced observations, in particular, self-observations. While triple-space based data integration is an established approach, our architecture relies on standard protocols and reliable data storage products, which is an advantage compared to most existing solutions.

We believe, however, that the central contribution of this work lies with its approach to systematic collection of human observations via instant messaging. To the best of our knowledge, this has not been implemented before. Beyond collecting health-related self-observations, as in USEFIL, this approach can be utilized in a variety of other applications including citizen sensing, opinion polling, and feedback collection. Compared to Web survey forms, an instant messaging based survey (via a smartphone) has a number of advantages. Questions can be asked "while they are hot" and conveniently responded with just one touch, questions can be personalized, questions that are asked (or not asked) next may depend on answers given to previous questions, as well as there is an option of a seamless handover from, e.g., a PC to a mobile device during a questioning session. An expected result is a higher response rate and a higher accuracy of responses.

Although the focus of this paper is on data collection and integration, as it is in PCS studies, the proposed architecture is grounded in an even wider view of [14], [15], which one of the authors of this paper co-originated. While PCS focuses predominantly on observing, i.e., sensing, [14] argued for a need to address the *device-software-human triangle* in a way allowing each of the three worlds to perform any of the basic computing functions: sensing, actuating, processing, and control. While focusing in this paper on the integration of machine sensing with human sensing, our architecture equally allows the integration of physical actuating with human actuating (e.g., sending a message to a person with a request to switch off unnecessary lights), software processing with human processing (e.g., sending a message to a person asking to translate a term), and software control with human control (e.g., providing a person with a list of possible courses of action and asking to select one).

In USEFIL, the work in 2014 includes conducting several-month-long pilot studies in Greece, UK, and Israel with real elderly subjects at their own homes. These pilots will provide a performance and usability evaluation of the system described in this paper as well as of other USEFIL products. Beyond USEFIL, our future work plans focus on further development and exploitation of the instant messaging survey approach in other applications. Some of these applications, e.g., feedback collection, are mentioned above.

ACKNOWLEDGEMENTS

This work is performed in the EU FP7 project USEFIL: Unobtrusive Smart Environments for Independent Living (Grant Agreement 288532).

REFERENCES

[1] A. Sheth, P. Anantharam, and C. Henson, "Physical-cyber-social computing: An early 21st century approach," IEEE Intelligent Systems, vol. 28, no. 1, 2013, pp. 78–82.

[2] A. P. Sheth, P. Barnaghi, M. Strohmaier, R. Jain, and S. Staab, "Physical-Cyber-Social Computing (Dagstuhl seminar 13402)," Dagstuhl Reports, vol. 3, no. 9, 2014, pp. 245–263.

[3] K. Teymourian, L. J. B. Nixon, D. Wutke, R. Krummenacher, and H. Moritsch, "Implementation of a novel semantic web middleware approach based on triplespaces," in Proc. Intl. Conf. Semantic Computing, 2008, pp. 518–523.

[4] M. Compton et al., "The SSN ontology of the W3C semantic sensor network incubator group," J. Web Semantics, vol. 17, 2012, pp. 25–32.

[5] A. Sheth, "Citizen sensing, social signals, and enriching human experience," IEEE Internet Computing, vol. 13, no. 4, 2009, pp. 87–92.

[6] E. I. Papageorgiou, A. S. Billis, C. A. Frantzidis, E. I. Konstantinidis, and P. D. Bamidis, "A preliminary fuzzy cognitive map - based desicion support tool for geriatric depression assessment," in Proc. IEEE Intl. Conf. Fuzzy Systems, 2013, pp. 1–8.

[7] N. Katzouris, A. Artikis, F. Makedon, V. Karkaletsis, and G. Paliouras, "Event recognition for assisted independent living," in Proc. 6th Intl. Conf. Pervasive Technologies Related to Assistive Environments. ACM, 2013, pp. 26:1–26:5.

[8] C. Cannuscio, J. Block, and I. Kawachi, "Social capital and successful aging: The role of senior housing," Annals of Internal Medicine, vol. 139, no. 5 part 2, 2003, pp. 395–399.

[9] P. Wu and C. Miller, "Results from a field study: The need for an emotional relationship between the elderly and their assistive technologies," in Proc. Intl. Conf. Augmented Cognition, 2005.

[10] P. M. Barnaghi, W. Wang, C. A. Henson, and K. Taylor, "Semantics for the internet of things: Early progress and back to the future," Int. J. Semantic Web and Information Systems, vol. 8, no. 1, 2012, pp. 1–21.

[11] J. Honkola, H. Laine, R. Brown, and I. Oliver, "Cross-domain interoperability: A case study," in Proc. Conf. Smart Spaces (ruSMART), LNCS 5764, 2009, pp. 22–31.

[12] R. Savola and M. Sihvonen, "Metrics driven security management framework for e-health digital ecosystem focusing on chronic diseases," in Proc. Intl. Conf. Management of Emergent Digital EcoSystems, 2012, pp. 75–79.

[13] F. Portet, M. Vacher, C. Golanski, C. Roux, and B. Meillon, "Design and evaluation of a smart home voice interface for the elderly: acceptability and objection aspects," Personal and Ubiquitous Computing, vol. 17, no. 1, 2013, pp. 127–144.

[14] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan, "Smart semantic middleware for the internet of things," in Proc. 5th Intl. Conf. Informatics in Control, Automation and Robotics, 2008, pp. 169–178.

[15] V. Terziyan, A. Katasonov, J. Cardoso, M. Hauswirth, and A. Majumdar, "PRIME: Proactive inter-middleware for global enterprise resource integration," Eastern-European Journal of Enterprise Technologies, vol. 51, no. 3/12, 2011, pp. 3–16.