Interacting with a Context-Aware Personal Information Sharing System

Simon Scerri[1], Andreas Schuller[2], Ismael Rivera[1], Judie Attard[1], Jeremy Debattista[1], Massimo Valla[3], Fabian Hermann[2], and Siegfried Handschuh[1]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland
[2] Fraunhofer-Institute for Industrial Engineering, Stuttgart, Germany
[3] Telecom Italia Labs, Torino, Italia
firstname.surname@deri.org,
firstname.lastname@iao.fraunhofer.de,
massimo.valla@telecomitalia.it

Abstract. The di.me userware is a decentralised personal information sharing system with a difference: extracted information and observed personal activities are exploited to automatically recognise personal situations, provide privacy-related warnings, and recommend and/or automate user actions. To enable reasoning, personal information from multiple devices and online sources is integrated and transformed to a machine-interpretable format. Aside from distributed personal information monitoring, an intuitive user interface also enables the i) manual customisation of advanced context-driven services and ii) their semi-automatic adaptation across interactive notifications. In this paper we outline how average users interact with the current user interface, and our plans to improve it.

1 Introduction

The development of a context-aware, proactive information management system relies on the following three core system requirements:

i) the extraction, resolution and integration of various types of distributed data in the personal information sphere [10][1];
ii) the realtime processing of personal data and event streams in order to detect situations benefitting from warnings/suggestions/automation;
iii) the design of an intelligent user interface that is able to non-intrusively but effectively support people with their day-to-day tasks.

Each of the above corresponds to realised components of the di.me userware's architecture, shown in Fig. 1. The first is embodied within the Personal Information Model (PIM): an ontology-based representation of the unified personal information sphere. A wide-variety of information is extracted from personal devices (including computers, gadgets and smartphones) and online sources (including social networks) [9]. In particular, the Personal Information Crawler targets retreivable resources, profiles and social

[1] This term refers to all of a user's digital information, including heterogenous personal identities and resources, stored on various devices as well as online accounts and services.

M. Kurosu (Ed.): Human-Computer Interaction, Part V, HCII 2013, LNCS 8008, pp. 122–131, 2013.

interactions belonging to both a person and their contacts, whereas the Context Extractor targets realtime personal and contact activities by way of various device (e.g. GPS positioning, environment, device usage) and virtual sensors (e.g. location, people tagged nearby). The extracted information is semantically lifted on top of a set of integrated ontologies[2] covering various domains. Following their extraction, entities are processed in order to perform cross-source resolution and integration, thus enabling di.me to be used as an entry-point for distributed personal information management.

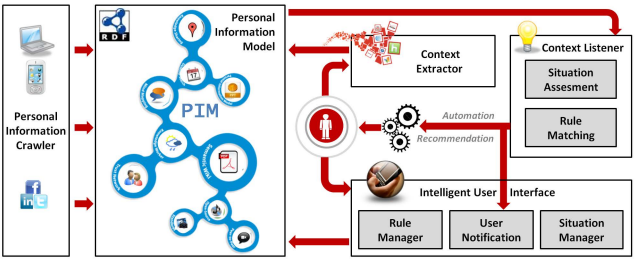

Fig. 1. The main concepts of our approach

Apart from providing improved personal information management, gathered PIM knowledge (including not only static information but also context data streams) is exploited by the *Context Listener* (Fig. 1) to pro-actively assist users with their knowledge-intensive tasks. The context listener provides two services: situation assesment and rule matching. The objective of the former is to automatically recognise recurring situations that have been marked by the user, whereas the objective of rule matching is to trigger context-driven rules, as defined by the user. These two services are enabled by a number of underlying techniques that can reason over the PIM knowledge.

The two above services depend on the user's interaction with di.me. To enable situation assesment, a person is required to initially mark situations of interest (which can subsequently be manually adjusted at all times). In contrast, the customisation of context-driven rules is targeted for manual setup by the user. Thus, an intelligent *User Interface* (UI) is a necessary di.me architecture component (Fig. 1), and the very success and usability of the userware relies on its ability to hide the complexity of the semantic processing from the users, while allowing them to take full advantage of its potential [5]. This places the di.me UI squarely within the category of intelligent UIs, i.e., "human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media" [7], where, in di.me, the said models are provided by the ontologies. As a complex system built upon comprehensive knowledge models, the di.me userware needs an appropriate way of interacting with the user. Therefore, as a second focus of this paper we will describe how the chosen UI design can, and to what extent, simplify the complexity of the existing models.

[2] The ontologies will be published under OSCAF: `http://oscaf.sourceforge.net/`

A survey of intelligent UIs categorises the observed functionality under five broad aims [4]. di.me's objectives fall under two of these categories. The first and primary purpose of the di.me UI is to "take over tasks from the user", reducing their workload by attempting to recognise their intent and propose or automise actions, thus allowing them to focus on other things. Specifically, the intelligent UI interacts with users to learn when automation/notification is most desired. In addition, the UI is also responsible for providing suggestions/warnings, or automating user tasks straightaway, as instructed by the user and recognised by the context-recognition system. The ultimate objective here is then to proactively suggest or even perform tasks for them according to their real life situation. A secondary aim of the di.me UI is to "support new and complex functionality", in order to ease the adoption and uptake of the di.me userware. As explained above, the di.me UI serves as an interface between the user's (mental model) and the system's (ontology representation) view of their personal information model. In addition, the implemented UI strives to make the user fully aware of the di.me userware's abilities by means of various interactive features, such as, context-dependent explanations accompanying intelligent suggestions, history-based functionality previews, etc.

In the remainder of this paper we describe the situation assesement and rule matching techniques, focusing on the user-device interactions across the intelligent UI. Before concluding, we provide an outline of future tasks and a comparison to related work.

2 Providing Context-Aware Intelligence

The Context Listener looks out for context changes registered in the PIM, resulting from either user (e.g. movement, device usage) or system (e.g. new file created, new contact added) activities, and corresponding to a respective PIM update (e.g. new location registered, new resource created). This information is then combined with other personal knowledge available, in order to detect i) recurring situations stored by the user, and ii) fire context-driven rules defined by the user. In this section, we give an overview of the techniques for situation assesment and context-driven rule matching, and demonstrate how the user's interaction with their UI can provide the desired intelligence.

2.1 Adaptive Situation Assesment

Context is any kind of information that can be used to characterise the situation of a person, as an entity [3]. Thus, to recognise recurring situations, di.me continuously monitors and interprets personal activities by exploiting the following three sources: Personal Devices, Services and Social Networks. Modern user devices (computer, smart phones, gadgets, tablets, etc.) provide a vast amount of personal activity context information. User attention monitoring constantly tracks what the user is doing on the device (e.g., foreground applications, web browsing activities, IM presence, etc.), whereas device-embedded sensor data is also monitored in the background to gather additional context (e.g geographical positioning, environmental conditions, network connectivity, etc.). Personal services registered on devices are monitored for additional context (e.g. weather, calendaring, emailing services). Finally, social networks are also targeted for context information, with both semi-structured (e.g. check-ins, tagging nearby people)

and unstructured (e.g. live post text containing references to nearby people, locations and current activities) being considered for a more complete snapshot of user activity context. This snapshot is constantly updated by the Context Extractor (Fig. 1) and stored within the user's PIM, as an instance of the purposely-engineered DCON Context Ontology[3], which we refer to as the *Live Context* [8].

Whenever they desire, users of the di.me system can mark personal situations (e.g. "Business Meeting", "Driving to Airport") through a specific button in the main UI dashboard. This will popup the new situation window, as shown in the (di.me-mobile UI, Fig. 2-a). Although a user can simply save the situation without looking at further options, the UI has been designed in a way that encourages them to provide additional information about the situations that they have just saved. This information is subjective and highly-dependant on the user's personal understanding of each individual situation. As shown in the example, people can remove irrelevant context items (e.g. the fact that the weather is Sunny), and assign different weights to relevant items (e.g. Jane is more linked to this situation than John is). The extremities of the 'weight bars' double as exclusion and necessity flags (e.g. if the user is in location DERI, then this situation can never recur). All these concepts are supported by different elements in the DCON ontology, which is also used to store situation instances. The shown UI design attempts to simplify the complexity of the DCON semantics so that non-technical people can fully-grasp them, and interact with the system to provide better situation representations.

To recognise a situation's recurrence, the Context Listener continuously compares its representation to the constantly-changing live context. Since both consist of structured RDF[4] graphs, a graph matching technique performs value matching at both DCON context element and attribute level. Therefore, even if two context elements (instances) co-occur in both graphs (e.g. Jane Doe is nearby in both the situation and the live context), context attributes (e.g. Jane Doe's actual detected distance) are compared to adjust the level of similarity accordingly. Similarly, if two context elements do not co-occur but have the same type (e.g. the situation graph refers to Jane Doe, whereas the live context graphs refers to John Doe), a partial match can still be detected based on the context attributes (e.g. they are both very near, or belong to a same group). A similarity function then determines which situations could be recurring, by taking into account all element-matching scores between each candidate situation and the live context, in combination with any user-defined element weights.

Users are notified of a situation that is matched with a high-enough confidence level (Fig. 2-b). For convenience, the situation matching bar always shows the three highest-matched situations at all times, even when initiated by the user. Through the bar, users can interact with the situation assesment results by dismissing or confirming the match. Live Context snapshots in the former case are stored as negative instances of the situation, whereas confirmed matches are stored as positive instances. Eventually, a situation (a DCON instance) will refer to multiple context logs. This way, the user's interaction with the UI results in the automatic increase/decrease of specific context element weights, resulting in a semi-automatic adaptive situation assesment service.

[3] http://www.semanticdesktop.org/ontologies/dcon/
[4] http://www.w3.org/RDF/

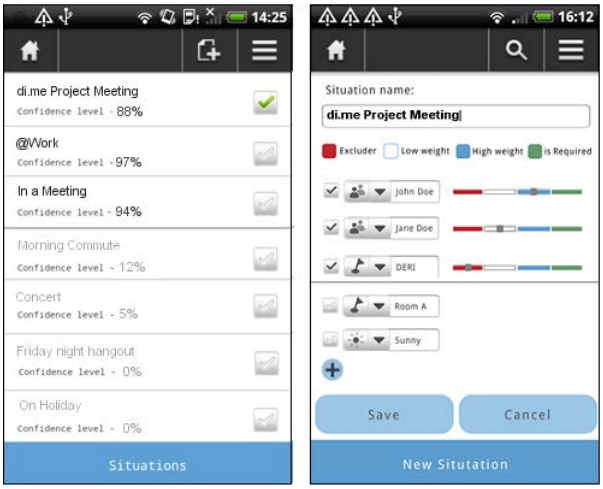

Fig. 2. Situation Notification (a) and Management (b)

2.2 Context-Driven Rule Matching

Context-driven rules (e.g., 'If Email from X received', *then:* 'Forward to Y', 'If in Situation "Business Meeting"', *then:* 'Change PhoneMode to Silent', 'If sharing items with untrusted contacts', *then:* 'Notify') can be created and managed through the *Rule Manager*. Inspired by approaches such as [6], user-defined rules in di.me are modelled on the Event-Condition-Action rule pattern concept: $if E[c_1, .., c_m] \implies [a_1, .., a_n]$ where the *event* E represents a rule that consists of a combination of conditions, trigerring one or more resulting *actions* a.

Rule conditions correspond to the wide-variety of concepts and information items that a typical person deals with through their devices (e.g., files, emails, contacts, locations, networks). Thus, rules are constructed out of items that can be stored in the PIM, based on concepts in the underlying ontologies and their attributes (e.g., an e-mail with a specific sender, a person with a low trust value, a meeting with certain people, etc.). To help the user with constructing personalised rules, the Rule Manager displays a selection of both i) ontology concepts (e.g., Person) and ii) known instances of these concepts (e.g., 'John Doe'). Whereas the latter identifies a specific instance that is known in the PIM, the former is equivalent to stating that any instance of the chosen concept will match the condition (e.g., any Person). The Rule Manager also allows users to specify *constraints* on top of conditions, based on the applicable ontology attributes. For example, a Person concept or a specific Person instance can both be constrained as follows: [Person].isAdded/.isRemoved (from the PIM), .isAddedToGroup, .isMentionedInMicropost, .approachesArea/.leavesArea (as detected by context sensors), .hasTrustValueChanged, .hasDataAccessGranted/.hasDataAccessWithdrawn (to any PIM resource).

Fig. 3-a shows three rules that have been created through the di.me-web UI Rule Manager. The latter enables users to create, modify and discard rules, as well as enable or disable them (tickbox on the left). Rules are constructed by selecting and constraining

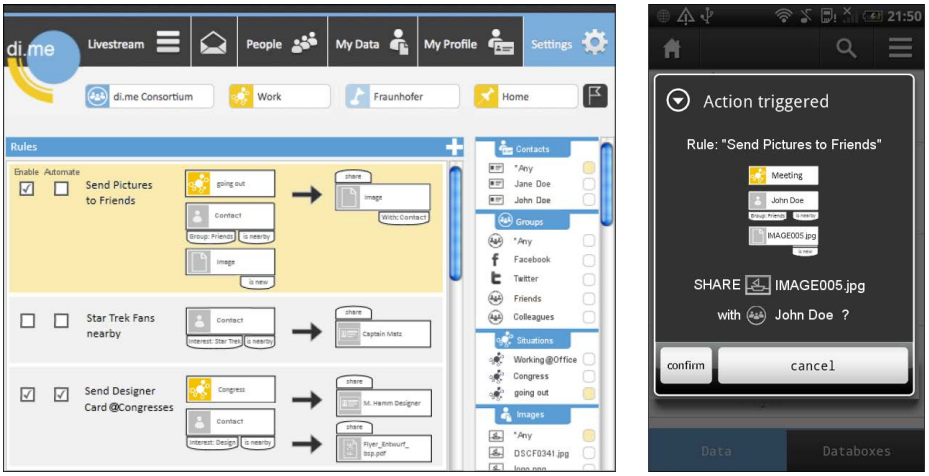

Fig. 3. Creating Context-driven Rules in the Rule Manager (a) and Rule Match Notifications (b)

PIM concepts and/or instances, as shown on the right hand side (e.g. contacts, groups, contacts, stored situations, images) . The first rule consists of three adjoined conditions: a specific user-created Situation, a Contact and an Image. The latter two conditions have been constrained to specify that the Contact needs to be nearby (isNearby) and a member of a specific group ('Friends'), and the Image needs to just have been created (isCreated). Therefore, the full rule can be described as: "When in Situation 'going out' *and* in the vicinity of any Contact(s) belonging to a Group 'Friends' *and* a new Image is created *then:* Share the Image with that Contact(s)".

Rule *events* trigger different types of actions such as *Notify*, *Post*, *Share*, *Change*, *Tag* and *Send*. A number of items are applicable to each of these actions (e.g. you can post a status message, link or photo; send emails and instant messages; change a person's trust level, a file's privacy level or your online presence message; share files and profiles etc.). Since the current di.me prototype only implements the sharing action, Fig. 3-a only demonstrates rules with this type of action.

Saved rules are translated to instances of the DRMO Rule Management Ontology[5]. The Context Listener monitors context-changing events in order to compare them against the defined rules. Context-changing events include: the creation of new PIM items (e.g. new e-mail received, new contact saved), the recurrence of known user activity context elements (e.g. user goes near a known location, connects to known wi-fi), and the re-activation of entire situations (as highlighted in Section 2.1). In order to match events to rule conditions, DRMO rules are mapped into SPARQL queries[6]. Stream event data is then matched against SPARQL triple patterns, pushing the rule onto a queue if a rule condition is matched. The context listener monitors future events, such that if all conditions are satisfied, it triggers the corresponding action(s) [2].

[5] http://www.semanticdesktop.org/ontologies/drmo/
[6] http://www.w3.org/TR/rdf-sparql-query/

The behaviour of the UI once a rule has been matched depends on the triggered action(s). Whereas notifications will simply result in dashboard system messages (including warnings and suggestions), other actions require user approval. For the featured sharing action, an interactive dialog box shows up, offering users a shortcut for sending matched items (images, other files, contact cards) to matched agents (contacts/groups). The dialog box includes an explanation of why the action is being propose. This includes presenting the user with matched variables for the trigerred rule, which in this case correspond to a matched Situation, a Contact (John Doe) matching the specified constraints (isNearby, Group:Friends), and an Image file matching the newly-created constraint (isNew). For this type of privacy-sensitive actions, the userware waits for an explicit go-ahead before performing the intended action (Fig. 3-b). For rules that have gained the user's trust, it is possible to override ('Automate' option shown for each rule in Fig. 3) the action confirmation box in order to perform the action(s) rightaway.

3 Future Enhancements

Through additional user-studies we intend to identify the best UI scheme to increase user interaction with the intelligent functionality provided by di.me. In particular, we will investigate non-intrusive techniques for i) avoiding 'cold start' userware problems by automatically detecting and suggesting candidate new situations and rules, rather than fully rely on the user for their initial setup; ii) providing improved justifications for system suggestions, warnings, etc.; and ii) enabling richer user feedback for the intelligent features, in order to speed-up the userware's learning process and user adaptivity.

The Situation Assesment technique will be enhanced to present candidate new situations to the user. For the purpose, the context matcher will closley monitor the changing context and look out for significant/sudden changes. The existing UI will therefore be extended to include new situation suggestions in the di.me dashboard, potentially sparing users the cognitive effort required to remember to mark new situations. At the moment, it is only possible for situations to be saved while they are still active. Since this is not always realistic, a context history visualisation will enable users to save important situation at a later time, by scrolling through a timeline and visually identifying it. Technically this is already possible due to automatic system logging, which also takes regular snapshots of a user's context for persistance.

Situation match notifications will be improved in order to provide justification as to why di.me detects that the situation has been reactivated. This could take the form of notification similar to the rule matching dialog box (Fig. 3-b), where instead of matched system/user events, the UI will display a list of context elements (e.g. people, wi-fi connections, time periods, running applications, locations, etc.) that are common to both the live context and the matched situation, together with their assigned weights. A further extension of this UI will also give the user an option to directly influence the situation assesment technique. At the moment, following a high match score, the user is able to either confirm (tick) or dismiss (untick) the detected situation (Fig. 2-a). As the proposed situation match notification extension will contain a list of detected context elements, the user could interact by removing irrelevant ones, or adjust their weight. Thus, the automatic weighting algorithm will be supplemented by direct user input, speeding up the learning process for the representation of the abstract situations.

The Rule Manager will be extended to support more complex user-defined rules, as supported by the DRMO ontology. Although the examples shown in Fig. 3 only employ the use of the *and* operator between both conditions and constraints, DRMO rules enable the use of additional logical operators. These include simple operators such as *or*, *and* and *negation*, as well as time-oriented operators like *succeededBy* and *preceededBy*. Supporting these operators in the Rule Manager will enable rules of type: "When in Situation 'Working at the office' *succeededBy* Situation 'Driving home from work' *then:* Change PhoneMode to Normal". Similarly, a more advanced UI will also support literal value constraints in addition to object attribute constraints. Here, users will be able to specify constraints on string or numeric values, through the use of simple operators such as $==$, $>=$, $<=$, $<$, $>$, as well as more complex ones like *contains* and *similar*. This will also enable rules of type: "New e-mail sent from 'John Doe' *and not* (subject *contains* 'di.me') *then:* Forward to 'Jane Doe' ".

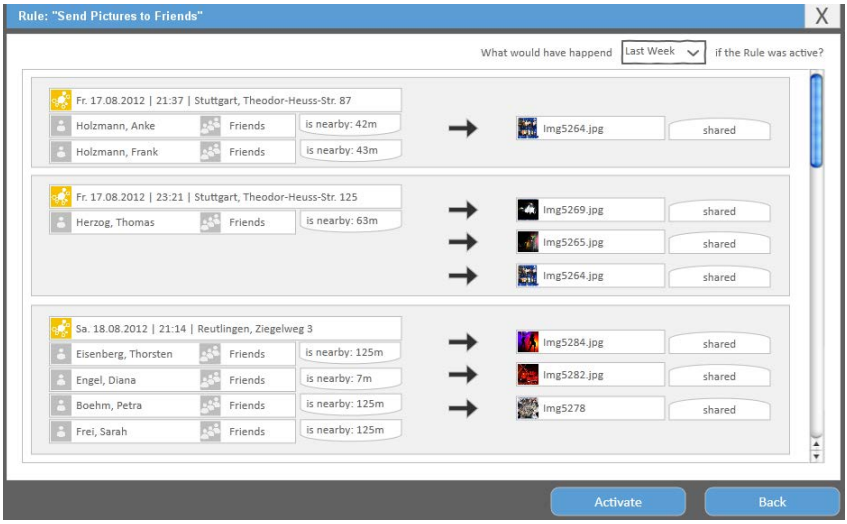

Fig. 4. History-based Rule Previews

A special feature being investigated regards the exploitation of the existent automatic system logging, in order to enable history-based rule previews. Here, users will be able to how a rule that they have just defined will function, based on their recent history. Thus, they will get a feel of how the rule will function in the future, by being shown all recent occasions which would have triggered the same rule. Fig. 4 shows instances of when the first rule shown in Fig. 3 would have been triggered in recent days, in this case showing which image would have been shared with whom, and why. This feature will easen the user's comprehension of created context-driven rules.

4 Related Work

Nokia Situations[7] is a context-aware smartphone application that adjusts one's phone settings based on the detected situation. Similar to di.me, Nokia Situations consist of user-configurable profiles (e.g. lunch, sleeping, watching television, etc.) that include a number of context parameters (e.g. location, time of day, time of week, wifi connectivity, etc.). Although the UI is able to let users interact with the concept of personal situations, the variety of context parameters supported by di.me through the DCON ontology remains unrivalled. Unlike di.me, the application does not adapt to the user by auto-adjusting situation representation during use. Nokia Situations also allow users to set different system actions (restricted to mobile phone configuration) based on a detected situation. In contrast, di.me can also fire rules based on other perceived system events (e.g. receiving an e-mail, adding a contact to a group).

Few of the comparable context-driven rule matching efforts enable users to personalise and manage their own rules. Beltran et. al. developed SECE [1] to enable user-defined rules for recommendations, based on open linked data services. Although rules are defined in an English-like 'rule language', the developers acknowledge the need for a UI that enables users having no technical skills to also create rules. In addition, di.me's adoption of ontologies as a standard knowledge representation format, as opposed to a system-dependant rule language, means that di.me rules can be processed, or re-used in external RDF-compliant systems.

Other efforts that are directly comparable to our personalisable rule manager UI are the on{x}[8] and IFTTT[9] services. on{x} also defines rules in terms of triggers and actions. Triggers are limited to the phone's sensors and abilities (e.g. GPS location, weather, wifi connections, battery). Like in di.me, a user can also select from various actions (e.g. notify, send text messages, invoke a Web service). Rules are created through the Website and pushed on to the user's phone, whereby they can only be disabled or deleted. Feature-wise, IFTTT is very similar to on{x}, providing a list of 'Channels' (e.g. Facebook app, Email service), triggers (e.g. new photo on Facebook, new email) and actions (e.g. post status message on Facebook, send an email) for setting rules through its interface. Both on{x} and IFTTT rules allow for the creation or the download of existing rules. The latter are termed 'recipes', and are a core feature of both services. Likewise, di.me offers a limited number of pre-defined 'generic' rules, for initial consumption by all users. The major contrast between on{x} and IFTTT is that whereas on{x} rule customisation requires Javascript proficiency, users can interact with IFTTT rules through an intuitive UI. Similarly, the di.me rule manager UI enables average users to visually select from a wide variety of events and their conditions, with the UI to DRMO instance translation being performed by the intelligent userware. In comparison to both on{x} and IFTTT, the di.me UI allows for more triggers, including detected personal situations, the creation/modification/deletion of known types of information elements (including documents, emails, persons, calendar events, etc.), and context changes detected by additional devices and sensors (e.g. online check-ins).

[7] http://www.pastillilabs.com/Situations
[8] https://www.onx.ms
[9] https://ifttt.com/

5 Conclusion

In this paper we introduce the main techniques behind a context-aware, intelligent information system that is able to automatically recognise recurring situations, and provide context-aware warnings, suggestions and automate system actions. In order to enable these functions, personal information is aggregated from personal devices and online sources, and combined in a personal knowledge repository with activities that are observed by various device and virtual sensors. The resulting machine-interpretable personal information is then utilised by an intelligent UI to enable the average device user to configure and manage i) personal situation detection and ii) personalisable context-driven rules. We also describe how the UI-user interactions are exploited to adapt the di.me userware to the user, by gradually improve the underlying techniques based on user feedback. Through descriptions of additional experimental features that are planned for future releases, we give an idea of the di.me userware's potential as a novel, intelligent personal information management system.

Acknowledgments. This work is supported in part by the European Commission under the Seventh Framework Program FP7/2007-2013 (*digital.me* – ICT-257787) and in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (*Líon-2*).

References

1. Beltran, V., Arabshian, K., Schulzrinne, H.: Ontology-based user-defined rules and context-aware service composition system. In: García-Castro, R., Fensel, D., Antoniou, G. (eds.) ESWC 2011. LNCS, vol. 7117, pp. 139–155. Springer, Heidelberg (2012)
2. Debattista, J., Scerri, S., Rivera, I., Handschuh, S.: Ontology-based rules for recommender systems. In: Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data (SeRSy12) at ISWC 2012 (November 2012)
3. Dey, A.K.: Understanding and using context. Technical report, Future Computing Environments Group, Georgia Institute of Technology, Atlanta, GA, USA (2001)
4. Ehlert, P.: Intelligent user interfaces: Introduction and survey. Technical report, Data and Knowledge Systems Group, Faculty of Information Technology and Systems, Delft University of Technology, Delft, Netherlands (2003)
5. Hermann, F., Schuller, A., Scerri, S., Thiel, S.: The digital.me user interface: an interaction concept for the management of personal information and identities. In: Proceedings of the 15th International Conference on Human-Computer Interaction(HCI 2013) (2013)
6. May, W., Alferes, J.J., Amador, R.: An ontology- and resources-based approach to evolution and reactivity in the semantic web. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3761, pp. 1553–1570. Springer, Heidelberg (2005)
7. Maybury, M.T., Wahlster, W. (eds.): Readings in intelligent user interfaces. Morgan Kaufmann Publishers Inc., San Francisco (1998)
8. Scerri, S., Attard, J., Rivera, I., Valla, M., Handschuh, S.: Dcon: Interoperable context representation for pervasive environments. In: In Proceedings of the Activity Context Representation Workshop at AAAI 2012 (2012)
9. Scerri, S., Cortis, K., Rivera, I., Fr, C.: Deliverable d03.03 data mining and semantic matching engine. Technical report, di.me Consortium (2012)
10. Scerri, S., Gimenez, R., Herman, F., Bourimi, M., Thiel, S.: digital.me—towards an integrated Personal Information Sphere. In: Workshop on Social Networks, Interoperability and Privacy by W3C, W3C Workshop, Berlin (June 2011)