Location based services for mobile users: a scenario based implementation account

Alberto Faro
Dept. of Electrical, electronics and Computer Engineering University of Catania
Catania, Italy
afaro@dieei.unict.it

Concetto Spampinato
Dept. of Electrical, electronics and Computer Engineering
University of Catania
Catania, Italy
cspampin@dieei.unict.it

***Abstract* - Many Location Based Services (LBSs) have been proposed, but they are developed mainly with a proprietary approach, i.e., they are specialized for managing specific tasks and are not able to make available the collected information to the services offered by other providers, thus limiting the potential benefits for the users. For this reason LBSs should be extended with a semantic layer able to provide the mobile users with the real time and off line information coming from the multitude of databases relevant for LBS activities, hopefully by using the same mobile device. Aim of the paper is to show how this can be accomplished in practice by integrating three methodologies in a single web architecture: a) a scenario based design methodology to implement the use stories of the mobile users, b) a semantic data management methodology to allow data integration, and c) a suitable JQuery Mobile based interface to allow the RoR web server to inform people on the available services independently on the adopted mobile.**

Keywords-component; Mobile Networked Applications; Intelligent Transport System; Scenario-based Design, Web-based Services;

I. INTRODUCTION

This Location based services (LBSs) are more and more relevant due to the pervasive diffusion of mobiles that allow metropolitan information centers to inform the citizens just in time on important events dealing with traffic, security and logistics. Although many LBSs have been proposed, they are mainly developed with a proprietary approach, i.e., they are specialized for managing specific tasks and are not able to make available the collected information to the services offered by other providers, thus limiting the potential benefits for the users and their own business opportunities.

This motivates why it has been proposed to extend the LBSs with a semantic layer able to provide the mobile users with the real time and off line information coming from the multitude of databases relevant for mobile activities, hopefully by using the same mobile device [1].

Aim of the paper is to show how this can be accomplished in practice by integrating three methodologies in a single web architecture, i.e. Ruby on Rails (RoR) [2], :

a) a scenario based design methodology to support a correct implementation of the use stories enacted by the mobile users;

b) a data management methodology to access the semantic layer where the information resident on the disparate databases relevant for LBSs stories is represented by a standard OWL format, and

c) suitable programming methods, e.g., HTML powered by Ruby, JQuery Mobile and Javascript, that allows the web server to interact in real time with the user mobiles, e.g., Iphone or Android, taking into account the user position.

In particular, when discussing the first issue we illustrate how two IS design methods, i.e. the story telling theory (STT) [3], [4] and the Behavior Driven Design (BDD) [5], may be integrated to implement safe and live web services using RoR. How RoR can be powered by means of the two main semantic ways to access the original data (i.e., centralized ActiveRDF-like API [6] vs distributed SPARQL [7]) is outlined depending on security and privacy issues. Finally, when illustrating the interfacing alternatives between the web server and the mobiles, some concrete solutions are sketched for possible reuse in similar contexts.

II. THE USE STORY BASED DESIGN TO IMPLEMENT EFFECTIVE RoR BASED WEB SERVICES FOR MOBILE USERS

Designing the information systems on the basis of the use stories is a key point to build user centered systems. This way of proceeding differs from the object centered approach mainly devoted to optimize the software architecture from the constructivist point of view.

In the nineties, SBD was mainly intended as a particular application of the Case based Reasoning (CBR) [8] to the information systems design. In the last decade, specific story based paradigms have been proposed more tailored to IS design, e.g. the story telling theory (STT) and the Behavior Driven Design (BDD).

In the following we outline how they work and can be used in a complementary fashion to verify the whole design of an user centered IS. Indeed STT powered by the CCS calculus [9] is suitable to verify that the specifications are safe and live, whereas BDD has been conceived mainly to test the software implementations against the specifications [10].

Both STT and BDD consist of a set of templates that should be used to specify the user behavior. According to STT a design should be partitioned in use stories that on their turn are subdivided in episodes. Each *use episode* consists of a temporal ordering of *actions*. The main feature of an episode is its atomicity, that implies that if an episode does not terminate successfully all the actions are cancelled and the system goes back to the state of the system before starting the episode.

The use episode represents the behavior of an actor, but to reach the episode goal, the main actor should interact with others. This is why in STT, we have introduced the notion of *scene* consisting of the parallel composition of all the use episodes concurring to reach the goal of the main actor. In SST such concepts are summarized by means of an episode template consisting of the following main categories: *what*, i.e., the goal of the episode, *assumptions*, i.e. the conditions (or the state) that enable the starting of the episode, *who*, i.e., the actors involved in the episode, generally they are the main character and one or two cooperating actors, *how*, i.e., the ordering in time of the actions too be executed by the actors to reach the goal, *what can go wrong*, i.e., the list of dangerous events that we intend to manage to avoid to restart the episode from scratch, *exception handling*, i.e., the ordering in time of the actions to recover the story.

The action flow is expressed in the section *How* by a sort of structured English. But, it should be expressed also by using CCS formulas, where the actions are denoted by $a_g!$ and $a_g?$; actions deal with a potential output or input exchanged by the actor A, through a communication gate g, with another actor. Operators ; and + indicate respectively the sequence or the choice between two actions, whereas the parallelism between two actions is indicated by vertical bar |. The execution of an action, e.g., $a_g!$, is given by the contemporaneous execution of the complementary action $a_g?$ by another actor. The parallel composition $a_g! \mid a_g?$ gives rise to an internal action i that is not visible from the outside. As an example, a scene S consisting of two concurrent episodes E_A and E_B may be expressed as follows:

$$S = E_A \mid E_B$$
$$E_A: s_{1A} ; d_1? ; a_g! ; b_g? ; goal! ; s_{2A} \quad E_B: s_{1B}; a_g? ; b_g! ; s_{1B}$$

where s_{1A} and s_{1B} are the conditions that allow the actors to start the execution of the episode. By using the CCS calculus we would obtain that:

$$S = (s_{1A\,,}\, s_{1B}); d_1? ; i ; i; goal! ; (s_{2A\,,}\, s_{1B})$$

i.e., when actor A is in state s_{1A} s/he is willing to execute the request d_1?, generated by the same actor A, of a certain service offered by another process B. The resulting parallel composition indicates that the cooperating episodes enacted by A and B, gives rise to a scene that it is not only safe, being not deadlocked, but also live since actor A reaches the goal with the support of actor B.

STT assures that the proof of correctness can be done even in case of very complex scenes consisting of several cooperating actors. This issue is outside the scope of the paper, but the interested reader may consult the Theory of Scenes and Interactions [11]. Of course, subdividing the specifications in stories allows the designer to verify the project not as a whole but story by story (or episode by episode), i.e., by a less costly verification approach.

It is very easy to pass from STT to BDD. Indeed, a BDD specification consists of a set of use scenario templates expressed by three main categories:

- *given*, i.e., conditions to allow the user to start the use case
- *when*, i.e., the first action of the use case
- *then*, i.e., the final action of the use case

As a consequence, it is straightforward to put into correspondence STT and BBD as follows:

assumptions <¯¯> given and how <¯¯> when; then.

Let us note now that, in general, the section *when* should deal with the first action of the episode and *then* with the final one, but one could adopt a finer description where *when; then* refer to a shorter sequence of actions. This correspondence allows us to rewrite the STT specs in BDD to prove that an implementation meets the specification *behavior* by *behavior*. In this way we have the possibility of proving what the engineers really need, i.e., that their software implementations satisfy the system specifications and that, on their turn, the implementations satisfy correctly the requirements agreed with the contractors.

However, these effective engineering tools would remain useless in absence of a programming environment oriented to implement a web service by stories. Fortunately, this weakness has been removed by using languages that follow the Model-Controller-View (MVC) programming paradigm, such as Ruby on Rails (RoR), i.e., a story based web programming environment based on *Models* to manage the data stored in tables as they would be objects, *Controllers* to implement story by story the interactions with the users, and *Views* to interact with the users with interfaces suitable for supporting the specific use stories.

Although many authors have deepened the Models section of MVC, the distinctive features of the MVC paradigm are the Controller and View sections. Indeed, as pointed out in the mentioned book on Rails [2], RoR aims at making feasible the user centered design through a stepwise refinement of an initial prototype derived from simple sketches of the user-system interactions. This "extreme" position is instrumental for stressing the importance of the user perspective in modern IS design and claims that complicated mathematical tools should be avoided. In principle, we agree with both these issues, but, as this section has demonstrated, STT/BDD is both formal and simple enough to prove that the software code has passed both the verification and testing phases.

Fig.1 demonstrates why designing by informal sketches may have many shortcomings. In particular, the sketch in the background is proposed in the mentioned RoR book to specify the user operations to purchase on line some books from a web store, whereas the geometrical lines aim at pointing out that the overall story should develop by the sequence of three episodes, i.e. *choose the books*, *fulfill the order*, and *collect the receipt.*

This clarifies that the story can stop at the end of each episode and may resume from where it has been left off, without restarting from scratch. Also, the superimposed diagram points out the processes involved and the tables required to support the document exchange.

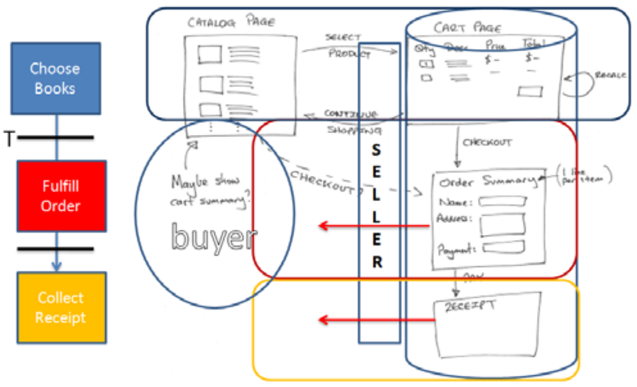


Figure 1. Specifications of a web store by sketches in the background, and corresponding STT diagrams in the foreground.

III. MAKING THE SEMANTIC LAYER PRODUCTIVE FOR THE MOBILE USERS

Although the Models section has not been conceived as the main feature of RoR, it is certainly a powerful environment to facilitate the object oriented management of the disparate databases of interest of LBSs usually expressed by tables, i.e., it hides the implementation issues of each database through objects, called ActiveRecords, that binds the tables to classes. This makes possible the data management by simple object formulas. To this aim, the records of the tables are extracted and recollected as objects. For example by the formula @parks = Park.all all the records of the table Park are made available to the programmer as objects @parks that can be programmed according to the object oriented language Ruby. For example, the field "name" of the Park table can be used as @parks.name.

Objects can be used either in the Controllers or in the Views. For example the following code refers to the View of the RoR Controller related to the story of informing the user on the best route to destination. In particular, all the destinations have been recollected by the object @posts in the Controller, and used within the *for-end* loop of the related View shown in fig.2 to produce the listing of the destinations from which the mobile user should select the preferred one. Ruby instructions are embedded in HTML code between two special symbols, i.e., <% and %>. The interested reader may find in [2] further details on how RoR may be programmed by using Ruby and Java script.

Although the RoR data management is stable enough, today the data management cannot be performed effectively by only the relational or object approach due to the relevant results obtained by the ontology engineering to integrate disparate databases available on the web. In the ontology approach, data are represented by a vocabulary of terms interrelated between them by subject-predicate-object relations, also known as RDFS-OWL [12].

```
    <div data-role="header">
      <A HREF="javascript:history.go(0)">Refresh</A>
      <h1>Destination Points</h1>
        <%= link_to 'Add', new_post_path, "data-icon" => "plus",
    "class" => "ui-btn-right" %>
      </div>
    <div data-role="content">
      <ul data-role="listview">
        <% @posts.each do |post| %>
        <li>
          <%= link_to post.title, post %>
         <%= link_to 'edit post', edit_post_path(post), "data-icon" =>
           "gear" %>
        </li>
        <% end %>
      </ul>
</div>
```

Figure 2. Software code of the user view of the fig.3b

As a consequence, RoR cannot be limited to bind tables to objects, but RDFS-OWL triples should be managed too. Such triples may be stored on a suitable LBS central server, or they may remain on their original sites, possibly together with the tables from which they are continuously updated. In the former case, we can adopt two main techniques to access the data: a) to use an API layer able to bind the triples to classes, e.g. ActiveRDF, so that the triples may be used as objects by means of the mentioned RoR object oriented formulas, or b) to process the response received from the server to a SPARQL query sent to the LBS server from RoR applications through the REST protocol service [2].

The former technique is suitable to access triples stored on a central server, whereas the latter may evolve towards a technique able to access triples resident on a distributed storage system. In the mentioned semantic LBS framework, both the centralized and distributed approaches should be adopted. Indeed, in the servers devoted to manage particular central DBs, such as account information, it is preferable to use an API layer based on objects rather than triples.

On the contrary, in the LBS applications where the server is devoted to inform the mobile users by taking advantage from all the information available on all the federated DBs, a distributed SPARQL query system should be adopted to access the data stored on their original sites rather than copying them on the central server. Indeed, it may be more effective to send the SPARQL query towards all the particular DBs and to process the responses received in SPARQL format rather than defining some super-object that integrates the objects obtained from the responses.

Such analysis might change substantially if one uses the current version of RoR, i.e., 3.X instead than 2.X. In fact, ActiveRDF and similar solutions should be are abandoned since they bind triples to the ActiveModels classes foreseen in RoR 2.X, whereas RoR is evolving towards a version, i.e., Rails 3.X, where the data layer is managed by Active Models and Active Relations.

As a consequence, it is needed a novel API layer that binds triples to RoR 3.X objects. Also, it is suggested to connect the RoR 3.X application to a distributed SPARQL service rather than using the outlined REST based query system that addresses separately the remote triple stores, thus avoiding the onerous phase of managing the changing configuration of the available DBs. Future works will clarifies if such novel techniques will make the semantic layer fully productive for mobile applications where real time information resident on disparate DBs play a key role to support the use tasks.

Concerning the main resources of an LBS ontology, e.g., parks, roads, intersections, pharms, we may take advantage from the proposals available in literature e.g., [13], [14] and [15], also called urban ontology or mobility ontology, to choose the best terms featuring the LBS vocabulary. Concerning the parameters affecting the individual terms (e.g., travel time, vacancies, and so on) they should be defined on the basis of the available sensing technologies, e.g., [16], [17], or from the available business information.

For this reason an LBS ontology should contain terms useful for both informing people on the available services and receiving measurements from the sensing devices. This allows the main LBS information center to inform mobile users about the availability of a *certain service in a certain urban area* (e.g. a parking or a pharmacy), and about *the service nearest to me* depending on the user location and on the current traffic conditions. Also, this would allow automated agents not only to operate on behalf of their users to conclude business transaction but also to process the collected data to obtain the parameters that characterize the time evolution of an LBS.

IV. Geolocation Interaction with Mobiles

The third key point of an effective LBS is the one of providing the mobile users with real time information by a suitable graphical interface [18] that meets the user visual attention [19]. This may be obtained by many programming languages depending on the particular mobile. But, avoiding the proliferation of the interface procedures is certainly a mandatory engineering requisite that may limit the use of many software frameworks. Hopefully, this requisite may be fulfilled by adopting the JQueryMobile (JQM) framework, whose software packages may be executed on the main mobile platforms and may be embedded into RoR 3.X applications together with the Ruby code and the Java scripts.

An example may clarify how this aim can be achieved in practice. The first deals with the classical request of the best route to reach the destination. Fig.3a shows the Google Maps obtained by a JQM procedure that detects automatically the current user position by executing within an RoR Controller the *navigator.geolocation* instruction. After measured these coordinates, the RoR Controller asks the user to choose the destinations from the ones listed on a specific View, e.g., the one shown in fig.3b, given by a JQM procedure following the lines indicated in http:// fuelyourcoding.com/getting-started-with-jquery-mobile-rails-3/. The user may choose either a fixed location (e.g., the Central Parking), or a location that depends on her/his current position (e.g., Close-By Pharmacy), or that is nearest to the destination (e.g., Close-By Downtown Parking). After clicking the relevant button, the mobile user will receive the suggested route to destination by another JQM procedure, e.g., fig.3c shows the map computed by reusing the software available at the page jquery-mobile-xample.html of the address http://jquery-ui-map.googlecode.com/svn/trunk/demos/.

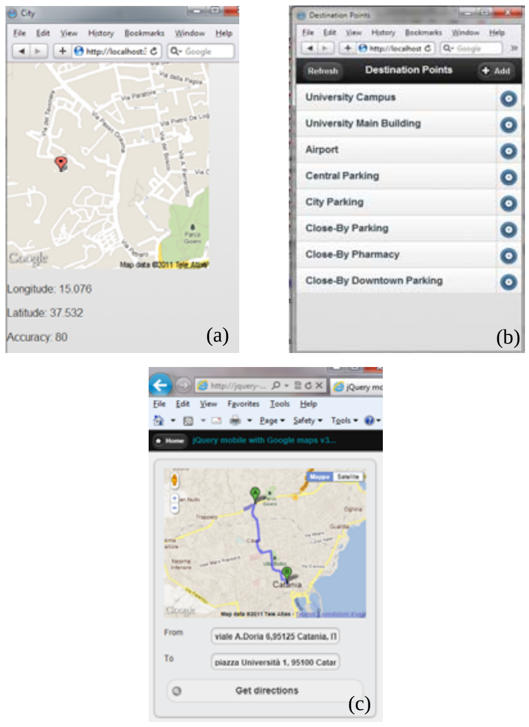
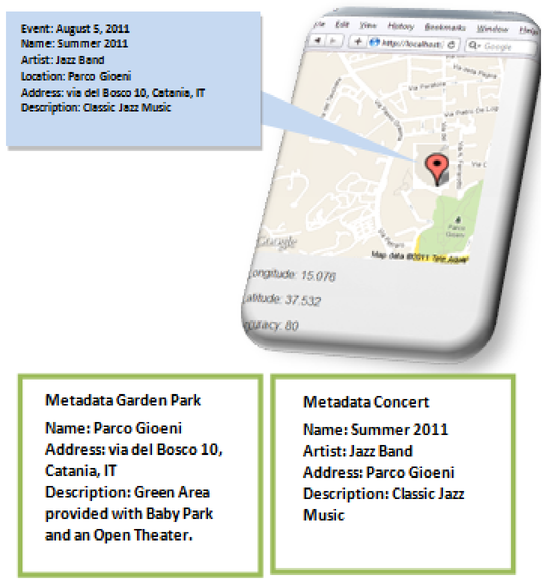


Figure 3. Current user position (3a), list of the destination points of user interest (3b), and route to destination computed by the LBS server (3c).

This example points out also that the user mobility may be improved greatly by this novel approach if the route to destination suggested by the LBS server is computed by using the current traffic conditions detected by a metropolitan monitoring system. In fact, this would allow the drivers to move towards the parking areas with vacancies that are nearest to their current position or to the destination depending on the traffic conditions. Also, this example shows that an RoR based LBS server powered by JQM procedures would allow the mobile users to carry out more powerful actions such as to buy products or to reserve a service on the fly.

This opportunity is further increased if the RoR LBS server makes use of the information available on all the metropolitan databases by using the RDFS-OWL approach outlined in the previous section. Fig.4 shows how joining, by a SPARQL query, the triples (or metadata) resident on two DBs, i.e., the ones related to the Garden Parks and to the Concerts, we may offer to the user a richer information on the metropolitan events on her/his mobile.


Event: August 5, 2011
Name: Summer 2011
Artist: Jazz Band
Location: Parco Gioeni
Address: via del Bosco 10, Catania, IT
Description: Classic Jazz Music

Metadata Garden Park
Name: Parco Gioeni
Address: via del Bosco 10, Catania, IT
Description: Green Area provided with Baby Park and an Open Theater.

Metadata Concert
Name: Summer 2011
Artist: Jazz Band
Address: Parco Gioeni
Description: Classic Jazz Music

Figure 4. The use of metadata facilitates the integration of different DBs and provides the users with more intelligible information.

IV. Concluding Remarks

The paper has illustrated how RoR 3.X may be powered by using RDFS-OWL technologies and JQM procedures to offer real time semantic services to the mobile users. In particular semantics allows the integration of the data resident on the disparate DBs available at metropolitan scale.

Also, semantics favors integration of the traffic measurements collected by the sensing system and makes more intelligible to the users the information provided on their mobiles. RoR makes also feasible to design applications according to the user centered design. This makes more clear for the user the procedures to follow to reach their goals. In particular, we have shown that such approach favors the modularity of the design and the traceability of the possible software bugs, thus confining the maintenance or revision procedures within specific portions of the specifications and the related codes.

Further works should be devoted to optimize the integration of resources resident on different DBs by using distributed technologies such as distributed SPARQL tools or distributed versions of the ActiveRDF-like approaches for supporting RoR 3.X distributed applications. Also, how integrating JQM procedures in RoR should be better understood to further improve the user-machine interface. Design memories consisting of software patterns [20] may help designers in repurposing the software, e.g., [21] and [22], thus reducing the development time [23]. Experimental results on the performance of a prototype under development in a project named *K-Metropolis*, partially supported by the Sicily Regional Government, will be available this year.

References

[1] A. Faro, D. Giordano, C. Spampinato, Integrating location tracking, traffic monitoring and semantics in a layered ITS architecture, IET Intelligenrt Transportation Systems, Vol.5, Issue 3, pp 197-206, 2011.

[2] M. Hartl, Ruby on Rails 3, Addison Wesley, 2011.

[3] A. Faro, D. Giordano, StoryNet : an Evolving Network of Cases to Learn Information Systems Design. IEE Proceedings SOFTWARE, pp.119-127, 1998.

[4] A. Faro, D. Giordano, Concept Formation from Design Cases: Why Reusing Experience and Why Not. Knowledge Based Systems Journal, vol. 11, n.7/8, p.437-448 ,1998.

[5] D. North, Introducing BDD, http://dannorth.net/introducing-bdd/

[6] E. Orel, R. Delbru, ActiveRDF: Object Oriented Semantic Web Programming, Proc. f the16th Int. Conf. on WWW. ACM, 2007.

[7] B. Quilits, U. Leser, Querying distributed RDF data sources with SPARQL, Proceedings of the 5th European semantic web conference on The semantic web, ESWC'08, 2008.

[8] S.L. Mansar, F. Marir, H.A. Reijers, Case-Based Reasoning as a Technique for Knowledge Management in Business Process Redesign, Electronic Journal on Knowledge Management, Volume 1 Issue 2, 2003 113-124 Academic Conferences Limited 2003

[9] R. Milner, A Calculus of Communicating Systems, Springer, 1980

[10] D. Chelimsky, et al., The RSpec Book, O'Reilly Vlg. Gmbh & Co.

[11] A. Faro, D. Giordano, A theory of interactions and scenes for user centered systems specification and verification, Proc. of the Asia Pacific Software Engineering Conference , APSEC '97. 1997.

[12] D. Allemang, J. Hendler, Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL, Elsevier Ltd, Oxford, 2011

[13] J. Teller, Ontologies for an Improved Communication in Urban Development Projects, Studies in Comp. Intelligence , 61, 1–14 2007.

[14] A. Faro, D. Giordano, A. Musarra, A. Ontology based intelligent mobility systems. IEEE SMC'03 Proc. Int. Conf. on Systems, Man and Cybernetics, Washington, 2003, Vol.5, 4334-4339. IEEE , 2003.

[15] Vilches Blázquez L.M, et alii, Towntology & hydrOntology: Relationship between Urban and Hydrographic Features in the Geographic Information Domain, Studies in Computational Intelligence, 2007, Volume 61, pp 73-84, 2007.

[16] A. Faro, D. Giordano, C. Spampinato, Evaluation of the traffic parameters in a metropolitan area by fusing visual perceptions and CNN processing of webcam based images. IEEE Transactions on Neural Networks, vol.19, issue 6, pp 1108-1129, 2008.

[17] A. Crisafi, D. Giordano, C. Spampinato., GRIPLAB 1.0: Grid Image Processing Laboratory for Distributed Machine Vision Applications. Proc. Int. Worshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE '08. IEEE, 2008.

[18] Giordano, D. Evolution of interactive graphical representations into a design language: a distributed cognition account, International Journal of Human-Computer Studies Vol. 57, Issue 4, pp 317-345, 2002.

[19] A. Faro, D. Giordano, C. Pino, C. Spampinato, Visual attention for implicit relevance feedback in a content based image retrieval, Eye Tracking Research and Applications Symposium (ETRA), 73-76, 2010.

[20] A. Faro, D. Giordano, Design memories as evolutionary systems: socio-technical architecture and genetics, IEEE Proc on Systems Man and Cybernetics, pp 4334 - 4339, 2003.

[21] C. Bizer, et al., Linked Data: Principles and State of the Art, 17th International World Wide Web Conference W3C Track @ WWW2008, Beijing, China 23-24 April, 2008.

[22] S. Dietze, H.Q.. Yu, D. Giordano, E. Kaldoudi, N. Dovrolis, D. Taibi, Linked education: Interlinking educational resources and the Web of data. Proceedings of the ACM Symposium on Applied Computing, SAC, 366-371, Trento, 2012.

[23] S. Ahmed, Encouraging reuse of design knowledge: a method to index knowledge, Design Studies, Volume 26, Issue 6, pp 565-592, 2007.