Modeling, Modeling, Modeling: from Web to Enterprise to Crowd to Social

Marco Brambilla and Stefano Ceri

Politecnico di Milano. Dipartimento di Elettronica, Informazione e Bioingegneria.
Via Ponzio 34/5. I-20133 Milano, Italy
marco.brambilla@polimi.it, stefano.ceri@polimi.it

Abstract. Data management is continuously evolving for serving the needs of an increasingly connected society. New challenges apply not only to systems and technology, but also to the models and abstractions for capturing new application requirements. In this paper, we describe several models and abstractions which have been progressively designed to capture new forms of data-centered interactions in the last twentyfive years – a period of huge changes due to the spreading of web-based applications and the increasingly relevant role of social interactions. We initially focus on Web-based applications for individuals, then discuss applications among enterprises, then we discuss how these applications may include rankings which are computed using services or using crowds; we conclude with hints to a recent research discussing how social sources can be used for capturing emerging knowledge.

1 Introduction

Long time ago, in the past century, the International DB Research Community used to meet for assessing new research directions, starting the meetings with 2-minutes *gong shows* to tell each one's opinion and influencing follow-up discussion. Bruce Lindsay from IBM had just been quoted for his message – very brief: "there are 3 important things in data management: performance, performance, performance". The oldest author of this paper had a chance to speak out immediately after and to give a syntactically similar but semantically orthogonal message: "there are 3 important things in data management: modeling, modeling, modeling".

Of course, if one compares the popularity of 3P and 3M in the data management scientific production, the balance is much in favor of 3P. Query optimization, indexing, parallel and distributed databases, cloud engines are much more popular than semantic models. Yet, we believe that performance is often attacked without a solid understanding of application needs, resulting in a brute force waste of energies – whereas more modeling could also lead to an overall better performing data system.

As a convincing example, we recall a consultant job for an anonymous Cefriel client[1], concerned with the overall quality of a very large relational database, originally designed for a commercial DBMS. By adopting systematic good modeling practices for improving the diagram readability and after deep semantic analysis with the designers, the original schema was progressively reduced from Fig. 1a to Fig. 1d – and only then implemented for performance. Our claim is that giving performance to the first schema would not solve its many problems of data redundancy and lack of orthogonality.

In this specific real-life case, an initial sub-optimal design had to be rectified though a number of modeling choices before even *thinking* to its performance; we conjecture that this occurs in many other real-life data-centered applications. Hence our "3M" motto. We argue that mastering "3P" and disregarding "3M" could be very dangerous, and therefore equal relevance should be given to data abstractions for semantics – e.g. in the form of high-level or abstract models – and to data structures for implementation – e.g. in the form of specialized persistent data structures or use of parallelism for performance.

We dedicated most of our research to 3M, by inventing new models and by applying modes to real-life scenarios so as to validate and use them. This paper is about model evolution in the last 25 years. Although we also worked on plain data modeling, we keep this aspect outside of our outline of this paper, and refer to Batini's article in this same book. We instead focus on the so called "emerging technologies" – although a 25-years-long period in ICT makes the emerging technologies at the beginning of the period almost obsolete at its end.

Thus, we concentrate on the following technologies: (a) the web, with its evolution throughout all the considered period – from exclusively desktop to mostly mobile; (b) the services, both in their interaction with web models and in their interplay for building search applications; (c) the crowds, and specifically the evolution from strict use of marketplaces such as Amazon Mechanical Turk to the adoption of social networks as sources of work; and (d) social sources themselves, seen as potential repositories of up-to-date knowledge.

This journey of course capitalizes on our results and is very much biased to emphasizing the 3Ms. We are aware that in certain cases some models apply to very few instances, possibly just to our own work. In other cases, however, 3M has been a successful vehicle for commercialization and standardization. Our work is traced by papers that appeared at WWW Conferences of the period, dedicated to WebML [17], to liquid queries over search services [4], to crowd-based search [3] and to knowledge extraction for social sources [9].

2 Web Modeling with WebML

Several researches have applied software engineering and Web engineering techniques to the specification of Web and multi-platform application interfaces and user interaction in broad sense. Among them, we can cite OO-HDM [26], WAE

[1] Hereby acknowledged for allowing us to publish the anonymized database schemas in Fig. 1; Cefriel is an IT center of excellence linked to Politecnico di Milano.

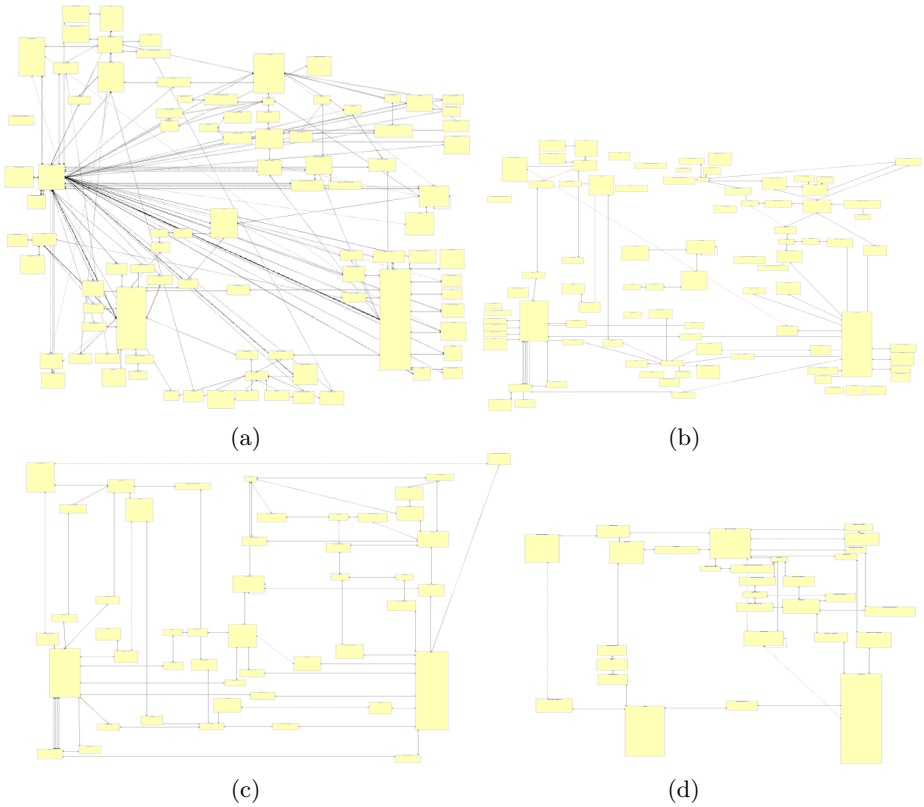

Fig. 1. Schema redesign in three steps. Step 1 eliminates from the diagram the tables which are only used as *active domains*, i.e. carry legal values for given domains. Step 2 isolates small *stars*, i.e. subentities carrying multivalued attributes. Step 3 eliminates *redundant information* from the schema and was the result of long discussions about data semantics.

[19], WebDSL [21], OOH-Method [20], WebML [16], RUX-Model [23], HERA [28], and rapid UI development [25] and modeling languages like USIXML [22]. Commercial vendors are nowadays proposing tools for Web development, like Mendix (`http://www.mendix.com`), Outsystems (`http://www.outsystems.com`) and Webratio (`http://www.webratio.com`). However, none of them has managed to become widely adopted in the software industry yet. For this reason, front-end development continues to be a costly and inefficient process, where manual coding is the predominant development approach, reuse is low, and cross-platform portability remains difficult.

WebML was an offspring of several EU-Funded project. It was first presented at the WWW Conference in 2000 [17] and then consolidated in a monography [18]. The specification of a Web application in WebML consists of a set of orthogonal models: the application data model (a standard Entity-Relationship model), one or more hypertext models expressing the navigation paths and the

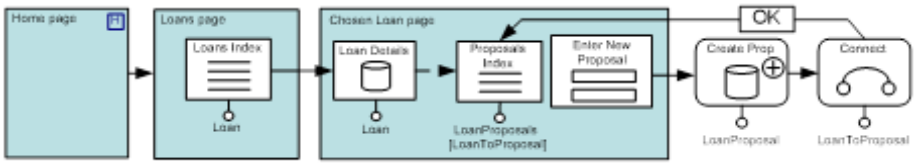

Fig. 2. Example of WebML hypertext

page composition; and the presentation model, describing the visual aspect of the pages. The presentation model is quite interesting, as it enables producing Web pages with the desired layout and look&feel for any rendition technology, but is outside the scope of this paper.

The hypertext model consists of one or more site views, each of them targeted to a specific user role or client device. A site view is a collection of pages (possibly grouped into areas for modularization purposes); the content of pages is expressed by components for data publishing (called content units); the business logic triggered by the users interaction is instead represented by sequences of operation units, which denote components for modifying data or for performing arbitrary business actions (e.g., sending email). Content and operations units are connected by links, which specify the data flow between them and the process flow for computing page content and for enacting the business logic, in reaction to users generated navigation events.

Consider for instance a simple scenario, whose hypertext is shown in Figure 2; users browse a Home Page, from where they can navigate to a page showing an index of loan products. When the user selects a loan from the index, he is taken to the Chosen Loan page, showing the loan details. In this page, a data unit, labeled Loan Details, displays the attributes of the loan (e.g. the company, the total amount and the rate), and is linked to another index unit, labeled Proposals Index, which displays the plan options. Then, the Enter New Proposal entry unit is used for data entry; the outgoing link of the Enter New Proposal entry unit activates a sequence of create and connect units, which respectively create an instance of the LoanProposal entity and connect it with a relationship instance to the Loan entity.

Syntactically, each type of unit has a distinguished icon and the entity name is specified at the bottom of the unit; below the entity name, predicates (called selectors) express conditions filtering the entity instances to be shown. WebML distinguishes between normal, transport, and automatic links. Normal links (denoted by solid arrows) enable navigation and are rendered as hypertext anchors or form buttons, while transport links (denoted by dashed arrows) enable only parameter passing and are not rendered as navigable widgets. Automatic links are automatically *navigated* by the system on page load.

WebML is associated with a page computation algorithm deriving from the formal definition of the models semantics, which describes how the content of the page is determined after a navigation event produced by the user. Page computation amounts to the progressive evaluation of the various units of a

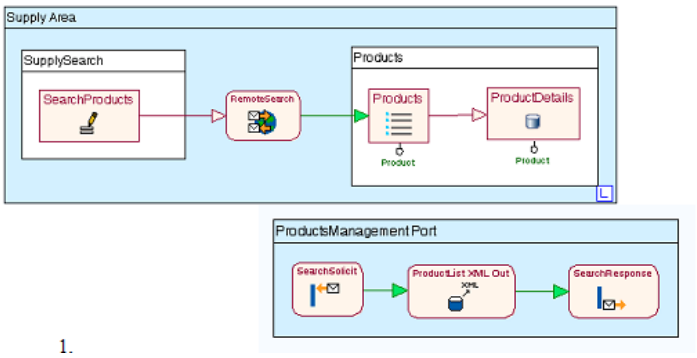

Fig. 3. Example of WebML hypertext model with invocation of remote service

page, starting from input parameters associated with the navigation of a link. This process implies the orderly propagation of the value of link parameters, from an initial set of units, whose content is computable when the page is accessed, to other units, which expect input from automatic or transport links exiting from the already computed units of the page. In WebML, pages are the fundamental unit of computation. A WebML page may contain multiple units linked to each other to form a complex graph, and may be accessed by means of several different links, originating from other pages, from a unit inside the page itself, or from an operation activated from the same page or from another page.

3 Enterprise Modeling with Services and Processes

Five years after we developed it, WebML evolved from model-driven web page generation to model-driven integration of applications within the enterprise. We discuss embedding of web services within WebML, and then use of WebML from within a generic enterprise workflow engine.

3.1 Service Integration

The first WebML extension is towards Service Oriented Architectures, with a focus on provisioning of well-designed services, usable across different Web applications [24]. Extensions to the hypertext model cover both service publication and consumption; service publication is expressed as a *Service View*, which is analogous to a site view, but contains specifications of services instead of pages. A service specification is denoted by a *Port*, which models the operations triggered upon invocation of the service.

Service invocation and reaction to messages are supported by specialized components, called Web Service units. These primitives correspond to the classical WSDL classes of Web service operations and comprise:

– Web service publication primitives: Solicit unit (representing the end-point of a Web service), and Response unit (providing the response at the end of a Web service implementation).
– Web Service invocation primitives: Request-response and Request units; they denote the invocation of remote Web Services from the front-end of a web application.

For instance, Figure 3 shows a hypertext that specifies a front-end for invoking a web Service and the specification of the web Service within a port container. In the former, the user can access the SupplySearch page, in which the SearchProducts entry unit enables the input of search keywords. The submission of the form, denoted by the navigation of the outgoing link of the entry unit, triggers a request-response operation (RemoteSearch), which builds the XML input requested by the service and collects the XML response returned by it. From the service response, a set of instances of the Product entity are displayed to the user by means of the Products index unit in the Products page; the user may continue browsing, e.g., by choosing one of the displayed products and looking at its details.

The lower part of Fig. 3 represents the service view that publishes the RemoteSearch service. The sequence starts with the SearchSolicit unit, which denotes the reception of the message. Upon the arrival of the message, an XML-out operation extracts from the service provider's database the list of desired products and formats it as an XML document. The service terminates with the SearchResponse unit, which returns the response message to the invoker .

3.2 Business Process Integration

With web services, the Web became a popular implementation platform for B2B applications, whose goal is not only the navigation of content, but also the enactment of intra- and inter-organization business processes. Web-based B2B applications exhibit much more sophisticated interaction patterns than traditional Web applications: they back a structured process, consisting of activities governed by execution constraints, serving different user roles, whose joint work must be coordinated. They may be distributed across different processor nodes, due to organizational constraints, design opportunity, or existence of legacy systems to be reused.

We extended our approach to cover business process based modeling [10], with a technique that exploits the BPMN notation for the description of the business requirements, and then maps them to hypertext model chunks that describe the user interaction of every task of the business process. The intuition is that the process progresses as the actors navigate the front-end, provided that the hypertext model and the process metadata are kept in synch. To this end, new primitives were added to the hypertext model, for specifying activity boundaries (namely activity areas) and process-dependent navigation (namely workflow links).

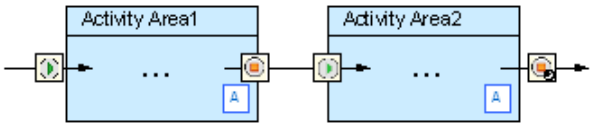

Fig. 4. Two activity areas and the start and end links that denote the initiation and termination of an activity

Figure 4 shows some of these primitives: Activity Areas denote groups of pages that implement the front-end for executing an activity; specialized links represent the workflow-related side effects of navigation: starting, ending, suspending, and resuming activities. Model transformations were used for translating a business process model into a skeleton of WebML hypertext model; a one-click code generator from the BPMN models generates running prototypes starting from the business processes, without the need of WebML modeling.

4 From WebML to the IFML Standardization and Commercialization

When we designed WebML, we thought that it was strategic to protect it through a US patent; WebML was implemented by WebRatio, a spinoff of Politecnico di Milano, which had unlimited rights of use of the patent. In the following ten years, the world of software tools drastically changed, and within and together with WebRatio we ended up promoting an open standard for enhanced hypertexts called Interaction Flow Modeling Language (IFML) [12], that was largely inspired by WebML; IFML was adopted in 2014 by the Object Management Group as an international standard after a 3-years adoption process. In the course of this operation, we had to give up on the protection of our ideas and completely change our approach [11].

4.1 IFML

From the technical perspective, IFML supports a much wider set of usage scenarios. Indeed, it aims at the platform independent description of graphical user interfaces for applications accessed or deployed on such systems as desktop computers, laptop computers, PDAs, mobile phones, and tablets. IFML adds to WebML several innovations: it increases separation of concerns, completely forbidding the integration of business logic into the user interaction specification; it defines a set of very generic concepts (the core of the language) which can be applied to any kind of user interface; it brings in the concept of event and asynchronous interactions; and it integrates seamlessly with UML and BPMN notations. The focus of the description is on the structure and behavior of the application front-end as perceived by the end user. Hence, with respect to the popular Model-View-Controller (MVC) model of an interactive application, the focus of IFML is mainly on the view part.

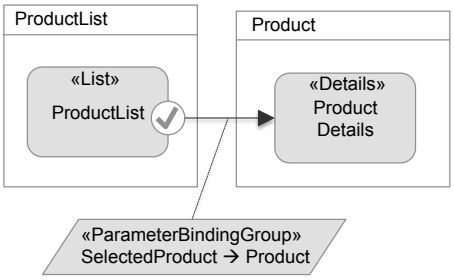

Fig. 5. Example of IFML model showing a list of products and the details view.

IFML models support the following design perspectives: (1) The *view structure specification*, which consists of the definition of view containers; (2) The *view content specification*, which consists of the definition of view components, i.e., content publishing and data entry elements contained within view containers; (3) The *events specification*, which consists of the definition of events (coming from user's interaction, application logic, or external agents) that may affect the state of the UI; (4) The *event transition specification*, which consists of the definition of the effect of an event on the user interface; and (5) The *parameter binding specification*, which consists of the definition of the input-output dependencies between model elements. Furthermore, IFML can be complemented with external models for connecting to any kind of content model (representing databases, ontologies, file systems or other resources) and any kind of dynamic model (describing the business logic behind the application front end).

Figure 5 shows a simple example of IFML diagram, where a starting page displays a list of products and, upon selection by the user, a target page shows the details of the selected product.

4.2 WebRatio

WebRatio is a commercial tool and company, born as spin-off of Politecnico di Milano, that has backed the development of WebML by progessively extending its supported features. Today, the WebRatio Platform[2] is a model-driven development tool based on IFML, which features two editions, respectively focusing on Web and mobile applications [1]. WebRatio provides an integrated environment for supporting the specification of IFML diagrams, including the view description, UML class diagrams for the information design, and optionally the integration with BPMN diagrams for the specification of business process aspects. It also includes a development environment for supporting the implementation of custom components and the layout template and style design environment, which allows the highest possible level of UI sophistication, thanks to full support of HTML 5, CSS and JavaScript based styling.

Based on the input provided through these environments, WebRatio applies a full-fledge model-driven development approach (as described a book of one

[2] `www.webratio.com`

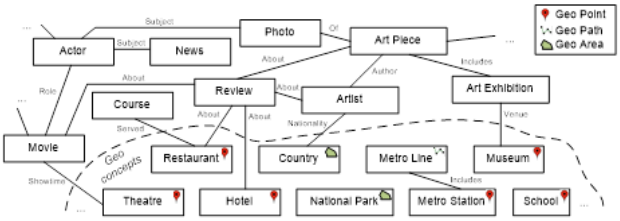

Fig. 6. Conceptual model with geo-referenced concepts

of the authors of this paper [6], which has become one of the reference readings in the MDD field), which provides model checking, full code generation, group–work support and lifecycle management. The generated code consists of: automatic cloud-deployed Java EE code covering both front-end of back-end of web applications for the Web version of WebRatio; and ready-to-deploy cross-platform mobile applications for the Mobile version of WebRatio. In the deploy, integration and coherency between mobile and web application is granted by a common modeling approach.

5 Search Modeling and Computing

With the start of Search Computing, an ERC-funded project (2008-2012)[3] our interest moved into the integration of search services, i.e. of services capable of extracting ranked responses [13–15]. In this context, modeling interest turns to understanding effective ways of combining services so as to extract only a few answers from them - top answers combine in creating query responses. Although search services can be part of arbitrarily complex applications, their typical usage is to answer queries, such as: *Who is the best doctor to cure insomnia in a nearby hospital? Where can I attend an interesting conference in my field close to a sunny beach?*; such queries are normally geo-referenced and dealing with distances is part of the ranking problem. The Search Computing project devised exploratory user interfaces, service registration tools, query configuration tools, and execution plan optimization techniques.

In particular, at the conceptual level, location-based resources are annotated with labels denoting their geometrical class (point, path, area) and grouped within a specific geo-concept region of the domain diagram. Fig. 6 shows a sample domain diagram including general entities and relationships and a geo-referenced region comprising entities such as: Museum, Restaurant, etc. Users can explore concepts by requesting for details of a specific entity or by moving to other, geographically or semantically related entity; we called such search paradigm a *liquid query* [4].

The exploratory search strategy evolves as follows. Users start by selecting one of the available entities, and submit a query to extract a subset of object instances. Among these, they select the instances they are interested in and then

[3] `http://search-computing.deib.polimi.it`.

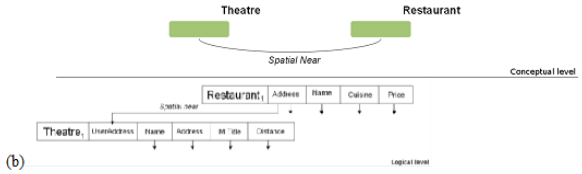

Fig. 7. Example of conceptual and logical service invocations

proceed by selecting the next entity to explore; the system retrieves connected object instances and forms several *combinations* with the previously retrieved ones; top-ranked combinations are displayed in ranking order. For instance, they can select a concert nearby their current location, then relate it to close-by transportation and parking facilities (spatial nearness), to other shows taking place the same night in town (spatial nearness and temporal proximity), performing artists (semantic relationships), etc.

The registration of geographical services and their relations used YAGO as reference knowledge base. At the logical level, spatial accesses are supported by services, e.g., GoogleMovies outputs movie shows ranked by distance from an input Location. Alternatively, spatial filters may be supported by ad-hoc services made available in the framework. Fig. 7 shows an example of how spatial concepts can be supported. At the conceptual level (Fig. 7(a)) the query searches for theatres close to restaurants. At the logical level, Fig. 7(b) it exploits the a service implementation of Theatre1 supporting distance, by matching the Address output of the service Restaurant1 to the UserAddress attribute of access pattern Theatre1. The nearness function is supported only by the Theatre1 service, therefore the relationship at the logical level is directed, indicating that accesses are possible by first selecting restaurants and then theatres, but not vice-versa.

This process can be particularly useful and efficient for geo-referenced objects that are searched on a mobile device. Fig. 8 presents a query combining concerts, restaurants, and hotels in San Francisco. By selecting one or more object combinations (Fig. 8(a)), users can prune the set of available options and look at the details of any object in the map (Fig. 8(b)), including related, non geo-referenced objects. Thanks to the conceptual model representation, the system clusters non geo-referenced items with the semantically closest geo-referenced item. Furthermore, starting from a given object, users can decide the exploration direction to follow towards other types of items (Fig. 8(c)) based on geographical relationships. In this case, additional objects appear in the map and contribute to the newly calculated combinations (and rankings), as shown in Figure 8(d). The exploration step can be iterated.

At any stage, users can move forward in the exploration by adding a new object to the query, starting from the available connections and from the objects that have been previously extracted. Users can also move backwards by excluding one of the entities from the query (e.g., removing hotels), or by deselecting previous manually selected objects. Backtracking at the level of individual conditions may help, e.g., in changing the restaurant choice from vegetarian to Japanese.

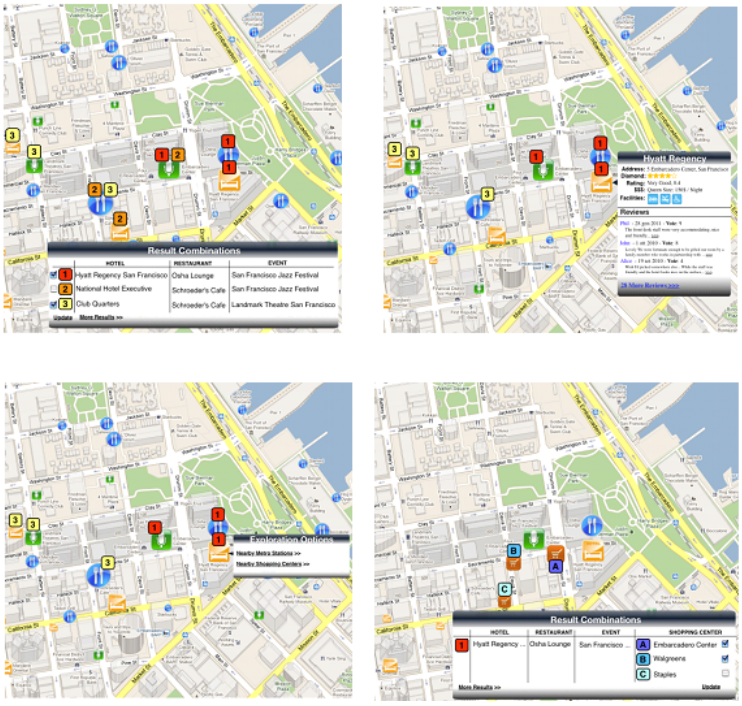

Fig. 8. (a) Visualization of best night-plan as combinations of hotel-restaurant-show; user selects combinations 1 and 3; (b) for the specific hotel Hyatt Regency of combination 1, a popup opens containing guest reviews; (c) a menu suggests exploration options relevant to the selected hotel metros and shopping centers; (d) user explores additional shopping centers near the selected hotel.

6 Crowd Modeling using CrowdSearcher

Amazon Mechanical Turk, the most widely used crowdsourcing marketplace, was created in 2005; but crowd-based computations became much more popular about 5 years later, when social communities (engaged though social platforms) were recognized as much wider and knowledgeable crowds. Humans were found more competent than machines in solving many tasks, ranging from simple ones (such as tagging images) to complex ones (such as finding optimal protein bindings in the 3D space); for what concerns search, the new trend of *asking the crowd* became popular: small *local crowds* could be selected based on geolocalization, expertise, memberships within special interest groups, or simply friendship – thereby complementing the results of search systems.

A that time, we developed CrowdSearcher, a model and tool for engaging crowds in the context of search queries [3]. Our complete paradigm, illustrated in Fig. 9, was alternating steps of search queries, using the search computing platform, and crowd-based queries, using a variety of social systems that could be invoked for providing human rankings. For instance, while planning a move

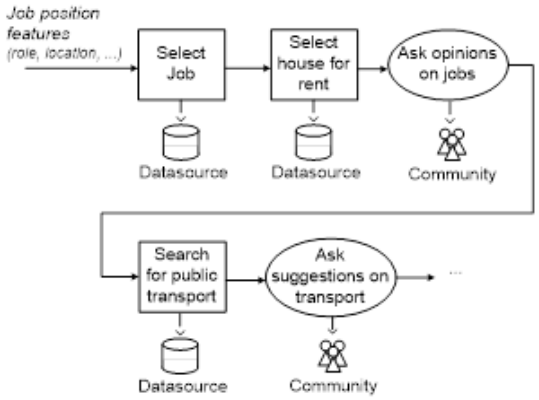

Fig. 9. Search session including two crowd-based search steps

Fig. 10. Example of intertwining crowd-based search within a search sessiom

to a new condition which was taking into account job availability, housing, and availability of public transport, experts could be asked to judge the alternative jobs or to comment about public transports.

An example of simple interaction for involving the crowd in suggesting jobs is illustrated in Fig. 10, where friends are asked to provide suggestions on the Facebook personal page of the person looking for jobs; in this case, the page displaying results displays for each result category the command `Ask the friends` which brings to an entry form where the user selects the social network and engagement process, including its duration; answers are automatically extracted and reproduced on a modified UI at the end of the period.

More in general, the input of a CrowdSearcher query QI is a triple $< C, N, S >$ where C is a data collection, N is a textual query expressed in natural language, and S is a collection of structured queries. Every component is optional. We next detail each component.

- C is an initial data collection which is proposed to the crowd for crowd-searching. For ease of description, we use the relational model, and therefore C is a collection of tuples. C can be sorted, in which case an attribute POS indicates the position of each tuple in the input sorting.
- N is a natural language query which is presented to the crowd. It can be mechanically generated, e.g. in relationship with specific structured queries, or instead be written by the user who starts the crowd search.

– S is a collection of structured queries that are asked to the crowd, relative to the collection C. Queries allow to express preferences about the elements of C, to rank them, cluster them, and change their content.

Preference queries correspond to typical social interactions (like, dislike, comment, tag); the other structured queries abstract simple and classical primitives of relational query languages which are common in human computation and social computation activities. The preference queries include:

– *Like* query, counting the individuals who like specific tuples of C.
– *Dislike* query, counting the individuals who dislike specific tuples of C.
– *Recommend* query, asking users to provide recommendations about specific tuples of C.
– *Tag* query, asking users to provide either global tags or tags about specific attributes of C.

The rank queries include:

– *Score* query, asking users to assign a (1..N) score to tuples of C.
– *Order* query, asking users to order the (top N) tuples in C.

The cluster queries include:

– *Group* query, asking users to cluster the tuples in C into (at most N) distinct groups.
– *OrderGroup* query, asking users to cluster the tuples in C into (at most N) distinct groups and then order the (top M) tuples in each group.
– *MergeGroup* query, asking users to merge N sorted groups producing a single ordering.
– *TopGroup* query, asking users to cluster the tuples in C into (at most N) distinct groups and then select the top element of each group.

The modification queries include:

– *Insert* query, asking users to add tuples to C.
– *Delete* query, asking users to delete tuples from C.
– *Correct* queries, asking users to identify and possibly correct errors in the tuples of C.
– *Connect* query, asking users to match pairs of similar tuples.

Our approach was subsequently integrated with a reactive paradigm so as to allow dynamic and continuous evolution of crowdsourcing strategies based on the response of the workers [5], by applying techniques largely inspired by the works on reactive databases and datawarehousing.

7 Using Social Content for discovering Emerging Knowledge

More recently, we focused on using big data produced by social interactions on platforms such as Facebook, Twitter, LinkedIn, Instagram. Some of our work was dedicated to using social sources for answering questions about Milano, such as understanding the languages being used in the various parts of the city [2], or the geographic spreading of Instagram posts after events such as the Fashion Week [7]. But we also tackled a more general research, consisting of using social content for capturing knowledge.

The most well-developed ontologies, such as DBpedia, Yago, the Knowledge Graphs in Google and Facebook, derive from structured or semi-structured, curated data. This process has involved huge efforts but had a huge payoff: DBpedia is now the crystallization point of linked data, while Google and Facebook saw the business value of this idea and have hugely invested in continuous and manual integration of databases for the development of knowledge graphs. So far, the effort of deriving knowledge has disregarder the contribution of social media, although social content has fueled the new discipline of Social Media Analytics [27], concerned with analyzing real world phenomena using social media.

We started a new research [8] targeted to discovering less popular items, those belonging to the long tail (e.g., the portion of the entity distribution having a large number of occurrences far from the "head" or central part of the distribution itself). Even the largest knowledge bases are largely incomplete for what concerns low-frequency data. It turns out, however, that knowing the long tail has a strong relevance, e.g., in e-commerce or search.[4] While high-frequency entities include well established brands, low-frequency data typically include *emerging* brands, those that have a small impact today but may have a high one tomorrow. The early discovery of low-frequency data and their ontological properties is thus a very interesting problem, with economic and practical implications in the innovation process.

Given these premises, our research focuses on the problem of *discovering emerging knowledge* belonging to the long tail, by extracting the low-frequency entities and relationships, with their attributes, from social content, thereby enriching existing domain knowledge [9]. We did so by using the methods for crawling social content and for entity recognition which are well established within social media analytics; our notion of ontology is broad, and includes classic cases, such as DBpedia or PubMed, but also any authoritative source of knowledge, such as the NY Stock Exchange Listings, or software projects in Github, or locations available in Open Street Map. These sources are used to define the ontological content of high-frequency entities.

We approach this problem with general, domain independent methods, but also with a well defined focus. We do not attempt at building full knowledge graphs, but rather we build small graphs, called *enriched domain graphs*, where

[4] The commercial success of Amazon and Google is due to their ability to discover goods or pages in the long tail.

the emphasis is on a given domain, and the enrichment is concerned with emerging concepts extracted from the long tail. Examples are: discovering emerging fashion designers (their identity / trends / brands)[5]; or discovering bloggers or narrative writers; or scouting emerging startups or products while they are becoming popular.

Domain knowledge is of course very useful in order to extract the relevant facts about the domain, e.g., high-frequency entities or relationships (thus, we know about Gucci or Prada) or structures from existing knowledge graphs (thus, we know that data about fashion designers can be linked to hubs such as fairs or magazines). We use such domain knowledge as the driver to select and organize relevant social content.

The method takes advantage of initial knowledge, that we call *seeds* and is typically provided by domain experts, to scout relevant *candidates* for the various kinds of emerging knowledge, extracted from social content, and ranked according to a variety of mechanisms, from syntactic to semantic ones, from information retrieval to machine learning, possibly helped by crowdsourcing; the first elements in the ranking are new concepts (e.g., entities or relationships), that can be validated by domain experts or, when confidence is sufficient, entered in the enriched domain graph.

In our future work, we plan to use social content to approach the dual problem of *detecting obsolete knowledge*, i.e., of knowledge that may have appeared at a given time but has not been confirmed as it has lost social confirmation. Examples in the medical domain include therapeutic options or theories about diseases which are very popular for a limited amount of time but then they either are ignored or confuted. In this case, we start from domain graphs, i.e., restrictions of knowledge graphs to specific domains, and we solve the dual problems of finding obsolete entities, relationships or attributes, and of discovering that certain types of the domain graph have lost relevance.

As an intellectual exercise, we are also interested in *detecting and confuting factoids*, i.e., studying the correctness of the domain graph. Specifically, one can search for factoids, i.e., assumptions or speculations that have been reported and repeated so often that they have become commonly accepted "facts," even though they lack any validity or truth. For instance, the belief that the Great Wall of China is visible from the moon is a factoid, as doing so would require a 17,000 times better eye resolution than we actually have.

8 Conclusions

Starting from the 3M motto, we presented several contributions to data-centered modeling of applications in the last twenty-five years. The lessons we learnt throughout our research is that **modeling must adapt to new concepts** and that focusing on the static aspects of conceptual schemas is not enough, as data has its own dynamics within constantly evolving applications. We started

[5] This problem is particularly relevant in Milano with its well-known fashion industry; it has been presented to us by the Fashion Design research group within Politecnico.

with hypertexts, added them services, workflows, and crowd-based computations which embed social contributions. Our recent work turned towards model construction, using social information.

Acknowledgement

We acknowledge many contributors to this work; among them, Piero Fraternali has a predominant role, being the main motor in all our works related to WebML and IFML. We also acknowledge the contributions of Alessandro Bozzon, Emanuele Della Valle and Florian Daniel, as well as a continuous interaction with Stefano Butti, Roberto Acerbis and Aldo Bongio from WebRatio.

References

1. Roberto Acerbis, Aldo Bongio, Marco Brambilla, and Stefano Butti. Model-driven development based on omg's IFML with webratio web and mobile platform. In *Engineering the Web in the Big Data Era - 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings*, pages 605–608, 2015.
2. Michela Arnaboldi, Marco Brambilla, Beatrice Cassottana, Paolo Ciuccarelli, Davide Ripamonti, Simone Vantini, and Riccardo Volonterio. Studying multicultural diversity of cities and neighborhoods through social media language detection. In *CitiLab, Papers from the 2016 ICWSM Workshop, Cologne, Germany, May 17, 2016*, 2016.
3. Alessandro Bozzon, Marco Brambilla, and Stefano Ceri. Answering search queries with crowdsearcher. In *21st Int.l Conf. on World Wide Web 2012*, WWW '12, pages 1009–1018. ACM, 2012.
4. Alessandro Bozzon, Marco Brambilla, Stefano Ceri, and Piero Fraternali. Liquid query: multi-domain exploratory search on the web. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 161–170, New York, NY, USA, 2010. ACM.
5. Alessandro Bozzon, Marco Brambilla, Stefano Ceri, and Andrea Mauri. Reactive crowdsourcing. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 153–164, 2013.
6. Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-driven software engineering in practice*, volume 1 of *Synthesis Lectures on Software Engineering*. Morgan & Claypool Publishers, 2nd edition, 2017.
7. Marco Brambilla, Stefano Ceri, Florian Daniel, and Gianmarco Donetti. Spatial analysis of social media response to live events. In *LocWeb , Papers from the 2017 WWW Conference Workshop, Perth, Australia. WWW Companion Volume, in print*, 2017.
8. Marco Brambilla, Stefano Ceri, Florian Daniel, and Emanuele Della Valle. On the quest for changing knowledge. In *Proceedings of the Workshop on Data-Driven Innovation on the Web, DDI@WebSci 2016, Hannover, Germany, May 22-25, 2016*, pages 3:1–3:5, 2016.
9. Marco Brambilla, Stefano Ceri, Emanuele Della Valle, Riccardo Volonterio, and Felix Acero Salazar. Extracting emerging knowledge from social media. In *Int.l Conf. on World Wide Web 2017*, WWW '17, page in print. ACM, 2017.

10. Marco Brambilla, Stefano Ceri, Piero Fraternali, and Ioana Manolescu. Process modeling in Web applications. *ACM Transactions on Software Engineering and Methodology*, 2006.
11. Marco Brambilla and Piero Fraternali. *Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML*. Morgan Kaufmann - The OMG Press, 2014.
12. Marco Brambilla, Piero Fraternali, and et al. The interaction flow modeling language (ifml), version 1.0. Technical report, Object Management Group (OMG), http://www.ifml.org, 2014.
13. Stefano Ceri and Marco Brambilla, editors. *Search Computing - Challenges and Directions*, volume 5950 of *Lecture Notes in Computer Science*. Springer, March 2010.
14. Stefano Ceri and Marco Brambilla, editors. *Search Computing - Trends and Developments [outcome of the second SeCO Workshop on Search Computing, Como/Milan, Italy, May 25-31, 2010]*, volume 6585 of *Lecture Notes in Computer Science*. Springer, 2011.
15. Stefano Ceri and Marco Brambilla, editors. *Search Computing - Broadening Web Search*, volume 7538 of *Lecture Notes in Computer Science*. Springer, 2012.
16. Stefano Ceri, Marco Brambilla, and Piero Fraternali. The history of webml lessons learned from 10 years of model-driven development of web applications. In *Conceptual Modeling: Foundations and Applications*, volume 5600 of *LNCS*, pages 273–292. Springer, 2009.
17. Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web modeling language (webml): a modeling language for designing web sites. *Computer Networks*, 33(1-6):137–157, 2000.
18. Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, and Maristella Matera. *Morgan Kaufmann series in data management systems: Designing data-intensive Web applications*. Morgan Kaufmann, 2003.
19. J. Conallen. *Building Web applications with UML*. Addison Wesley, 2002.
20. Jaime Gómez, Cristina Cachero, and Oscar Pastor. Conceptual modeling of device-independent web applications. pages 26–39, 2001.
21. Danny M. Groenewegen, Zef Hemel, Lennart C. L. Kats, and Eelco Visser. WebDSL: a domain-specific language for dynamic web applications. In Gail E. Harris, editor, *OOPSLA Companion*, pages 779–780. ACM, 2008.
22. Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Victor Lopez-Jaquero. USIXML: A language supporting multi-path development of user interfaces. In *Engineering Human Computer Interaction and Interactive Systems*, volume 3425 of *LNCS*, pages 200–220. Springer, 2005.
23. Marino Linaje, Juan Carlos Preciado, and Fernando Sánchez-Figueroa. A Method for Model Based Design of Rich Internet Application Interactive User Interfaces. In *Proceedings of International Conference on Web Engineering, July 16-20, 2007, Como, Italy*, pages 226–241, 2007.
24. Ioana Manolescu, Marco Brambilla, Stefano Ceri, Sara Comai, and Piero Fraternali. Model-driven design and deployment of service-enabled web applications. *ACM Trans. Inter. Tech.*, 5(3):439–479, 2005.
25. Arne Schramm, Andre Preussner, Matthias Heinrich, and Lars Vogel. Rapid UI development for enterprise applications: Combining manual and model-driven techniques. In *Model Driven Engineering Languages and Systems*, volume 6394 of *LNCS*, pages 271–285. Springer, 2010.

26. Daniel Schwabe, Gustavo Rossi, and Simone D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Proc. Hypertext'96*, pages 116–128, 1996.
27. Stefan Stieglitz, Linh Dang-Xuan, Axel Bruns, and Christoph Neuberger. Social media analytics. *Business & Information Systems Engineering*, 6(2):89–96, 2014.
28. Richard Vdovják, Flavius Frăsincar, Geert-Jan Houben, and Peter Barna. Engineering Semantic Web Information Systems in Hera. *Journal of Web Engineering*, 1(1-2):3–26, 2003.