Utilizing Open Content for Higher-Layered Rich Client Applications

Monika Steinberg, Jürgen Brehm
Institute of Systems Engineering - System and Computer Architecture
Hannover, Germany
[steinberg, brehm]@sra.uni-hannover.de

Abstract - **Accomplishing user interoperation and standardized web techniques is a promising mixture to build a next generation of web applications in the currently arising Social Semantic Web. Increasing heterogeneous Open Content is an ongoing trend. Generic concepts for higher-layered reuse of the arbitrary information overload - mentioning the Internet of Services - are not covered very well yet. For further, directed use of distributed services and sources, inquiry, interlinking, analysis, machine- and human-interpretable representation are as essential as lightweight user-oriented interoperation and competency in handling. In the following we introduce the *qKAI application framework* (qualifying Knowledge Acquisition and Inquiry) [1] - a service-oriented, generic and hybrid approach combining knowledge related offers for convenient reuse and tweaking them with interaction for improved access. qKAI aims at closing some residual gaps between the "*sophisticated*" Semantic Web and "*hands-on*" Web 2.0 enabling loose-coupled knowledge and information services focused on knowledge life cycles, learning aspects and rich user experience. Overall, in qKAI Open Content is boosted as an inherent part of higher-layered, lightweight applications in knowledge and information transfer via standard tasks of knowledge engineering and augmented user interaction. We introduce the *qKAI hybrid data layer* - a minimalistic data model with maximized depth - implementation results and some lessons learnt. We discuss the Semantic Web query language SPARQL critically to enlighten its limitations in current web application practice. Acquiring resources and discovering the Web of Data is a massively multithreading part of the qKAI hybrid data layer which serves as basis for further knowledge based tasks. Built upon this data layer, social educational gaming is instanced to simplify interoperation, to spread knowledge in a handy way and to enhance users' collaboration with Open Content. Attendance is increased through game-based, incentive arrangements following *Rich Client* paradigms. *Long-term objective* is to establish Open Content in information and knowledge transfer as utilized knowledge base.**

Keywords: Open Content; Social Semantic Web; Knowledge Engineering; Rich Clients.

I. INTRODUCTION

Currently the borders between Semantic Web and Web 2.0 become fluid more and more and let us create new synergies in the Web 3.0 [2] or also called the Social Semantic Web. The combination of social user involvement, employing desktop-alike rich interfaces (RIA [3]), and the Semantic Web with technologically oriented operability for data representation and processing is a promising conceptual basis to solve two pending problems. On the one side, there is still a lack of lightweight user participation in Semantic Web contexts because of handling hurdles and missing fancy interoperation ability. On the other side, there are claims for less trivial and more unitary content in Web 2.0 contexts. Currently DBpedia [4] and Freebase [5] start bringing these efforts together by offering collaborative content collection, creation, refinement, or semantic interlinking to increase Open Data that is well interpretable by humans and machines. Twine [6] is another semantic knowledge-enhancing platform, but does not offer its content with open access yet.

Metadata is an important factor for analyzing and categorizing content. In case of missing metadata, automated and manual annotations are approved workarounds to get information about the information while deriving useful knowledge out of it. Conclusions about information quality (e.g., provenance, reputation, timeliness, correctness) are important for further deployment in knowledge transfer scenarios and can be deduced out of metadata analyses and further interactive assessment.

We have to develop enjoyable interoperation scenarios that permit interactive knowledge transfer and learning. We see facilitating access to Open Data by intuitive learning interaction concepts as a promising combination to increase Open Knowledge and to prepare it for further learning purpose. Knowledge transfer is in contrast to learning a non-linear process. Learners are able to move free in the created environment and may decide on their own which learning order to take. Further on users are embedded and actively involved to influence learning sequences. Proved learning concepts have to be active, self-controlled, constructive, situative and social following successful didactic concerns [7]. Systematically linear and non-linear learning scenarios will be realized in the qKAI project [1] to allow different interaction types like exploring, questioning, answering or annotating.

Also fundamental are incentive and motivation of the users to interoperation and collaboration. Next to common methods for annotating and exploring data, using questionnaires and data browsers, we see especially knowledge games as motivating way to implicitly inquire, analyze and annotate content while knowledge is interceded. Well-designed gaming flows can impart handling of suspenseful information in an easy understandable manner to the user. Open Knowledge, that is well comprehensible for its users and machine-readable, increases this way. Newly developed learning interaction services and enriched content should be tied up with conventional Learning Management

Systems (LMS) and learning standards (LOM, SCORM, IMS/QTI [8]).

Obviously, there are many different tasks to perform, to utilize arbitrary available Open Data for higher-level, extensible and standardized applications with rich interoperation for knowledge transfer and learning. Our research showed that there are several knowledge bases, services and software components available that are required for sub tasks of qKAI. Therefore, the challenge is to merge and expand existing APIs, frameworks, autonomous services and distributed sources to perform our jobs here. According to C. Schroth and T. Janner [9] we see the relation of our needs to service-oriented software design (SOA): "*The first major analogy between product design in the fields of Web 2.0 and SOA is the notion of reusing and composing existing resources. Both concepts let users reuse, remix, and enrich existing resources and components to new and potentially higher-level applications. The second commonness is the affinity to collaboration and coupling of remote resources or services. Both Web 2.0 and SOA applications enable the loose coupling of distant and possibly heterogeneous resources. A third apparent resemblance between Web 2.0 and SOA is the shared principle of agility and the support of permanent structural change.*" [9]

Long-term objective is to embed different types of services (atomic, simple and composite services) in qKAI for systematically utilizing Open Data and enhancing Open Knowledge. Design concepts from service-oriented and mediator-wrapper-based information systems [10] are applied in the system specification of the qKAI framework. We identified three main service categories and packaged them in three service bundles as interaction, representation and discovery manager in a mediation layer (see Figure 2). To keep the system structure comprehensive and easy extensible we take a 4-tier-layer concept paired with Rich Client MVC2 paradigms to structure and model desired service managers and types.

A. *Structure of this contribution*

First, we introduce some further background. In Section 2 follows what we see as prerequisite to utilize Open Content for higher-layered applications. Section 3 gives an overview of the qKAI application frameworks' system design. Section 4 offers some more details concerning the qKAI hybrid data layer as one system level of the 4-tier design. Section 5 shows further services and components, Section 6 exemplifies use cases and further application scenarios. At least this contribution ends up with a conclusion and future work in Section 7.

B. *Resources and Open Content*

Open Content is interpreted in qKAI following the Open Knowledge specification "*Defining the Open in Open Data, Open Content and Open Information*" by the Open Knowledge Foundation [11]: "*A piece of knowledge is open if you are free to use, reuse, and redistribute it.*" qKAI adds processing differentiation between Open Data, as raw input information and Open Knowledge, which represents qualified information – checked or enriched yet. The semantic Web of Data (RDF stores and ontologies) and User Generated Content (Wikis, communities, Blogs) stand by and grow up in structured, unstructured and semi-structured manner. DBpedia offers an extensive knowledge base in RDF format [12] (generated out of Wikipedia content), allows semantic browsing and detailed thematic inquiring by applying SPARQL [13] queries for refinishing and further assignment.

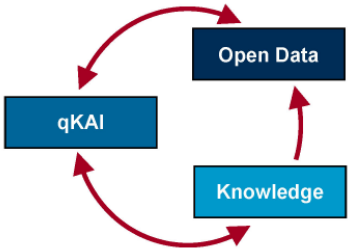

Figure 1. qKAI conept scheme to derive knowledge out of Open Data

RDF aims at the unique description of entities, their relations and properties on the internet [14] according to a standardized schema. These are the resources or "*things*" if we talk about the renewed "*Internet of Things*". The access to resources is always carried out using representations. One resource can have several representations like HTML, RDF, XML or JSON.

Open, shared databases like Freebase offer a free API to reuse its content by its own Metaweb query language (MQL) [5]. Relational databases can be easily converted into the Web of Data embedding existing components like the D2R server [15]. Additionally, many unstructured sources like HTML sites or PDF files do not apply to machine-interpretable web concepts yet. Serializing this data to standardized formats with open access is a first step towards enhanced machine and user interpretability. Aperture [16] and Virtuoso Spongers [17], for example, enable comprehensive solutions for these tasks. In case if more text engineering is needed, there are comprehensive solutions for standard Natural Language Processing (NLP) tasks (e.g., by OpenNLP [18]) to perform sentence detection, NER (Named Entity Recognition), POS (Part-Of-Speech) tagging or even semantic chunking.

C. *Quality of content*

Especially, if enabling User Generated Content (UGC) that is not authored by experts yet for knowledge transfer scenarios, the question of contents' quality arises. Therefore, we developed a **three level model** to handle different aspects of quality. Metadata can be seen as a quality feature [19]. The more metadata we are snapping, the better we get to know the content. There is no absolute quality, but we can compare resources with each other (Open World Assumption) and weight them based on the amount and structure of metainformation. Enrichment of a resource happens in a corresponding qKAI URI by semantic interlinking. One example is a domain ranking visualized as tag clouds to show from which domain we get the most information right now. First level criteria contain metadata directly included in a resource like format, timeliness, author, provenance or language, which can be automatically detected. Second level criteria are determined through user interaction, which helps to enhance semantically correctness. Regarding factual knowledge like "*Berlin lies at the Spree*"

or "*Hanover is the capital of Lower Saxony*", we see user rating and ranking following the established Web 2.0 manner as an effective solution to mark wrong content and to rank valuable or popular content systematically. Next to this crowd sourcing community approach, we offer role- and level-based quality control mechanisms. Lecturers earn rewards while rating and creating educational resources out of Open Content; students earn rewards while answering questions, managing gaming tasks, exploring further content or ranking their favorites. Gradually content can be qualified this way. Resources are marked following their quality level as reviewed, proofed or not yet qualified to enable embedding in different levels of knowledge transfer and learning. Third level criteria are inferred employing Natural Language Processing to detect some more information hidden inside a resource.

D. Interaction in the Social Semantic Web

We are looking for some innovative interaction to deploy Open Content for further purpose. The state of the art shows for example DBpedia mobile [20] that combines Wikipedia entities with images and places them on a map. At Revyu [21], we can rate and rank everything. For selection and requesting Open Data in RDF format there are SPARQL query tools available. Most of them are not very intuitive but technical. Altogether, we currently can **search, display, group, edit and annotate** content on the web. However, there is little cumulative advantage and not much incentive for the user to interact and to deduce new knowledge.

E. Games with a purpose, social and educational gaming

The idea of combining social networking, gaming and rating is not new. As far as we know, there are no applications available adding strong focus on knowledge and learning to it. Available social games do not rely on standardized Open Content or sustainable concepts. Gaming content is explicitly and laboratory created manually for every single game. Generic approaches to build an ongoing social knowledge network based on available content are still missing. Embedding gaming into superior learning structures regarding learning management standards, e-learning infrastructures and the Internet of services seems to be not mentioned so far. Different to other gaming approaches content creation in itself is part of our game-based learning concept. Players get the ability to change the view of relevant resources. For example, text and multimedia is presented as extracted chunks of information or image having knowledge-snack concepts on mind to enhance understanding and portioning not to overburden the user. Text sections out of articles are passed to the user and he has to guess the context. Sights are presented in detailed zoom view to let users guess them. Locations have to be placed at the right position on a map.

Luis van Ahn introduced crowd sourcing to gaming with ESP game or reCAPTCHA [22]. Guess-the-Google is a term-guessing game based on Google search results or images [23]. Scoyo [24] offers a game-based learning platform for kids, but does not deal with Open Content or Open Access. All over, there are no generic game-based learning concepts available regarding web standards and learning management based on Open Content. Additionally there are some commercial, social gaming applications like Playfish, MegaZebra or Zynga [25] with casual character. They are often embedded into Web 2.0 platforms like Facebook or MySpace [26] to increase participation.

Open Content is a huge knowledge base, but there are missing augmented interaction abilities to confront users with Open Knowledge bit by bit in enjoyable manner (knowledge-snacks, casual games). We can read Wikipedia articles or watch some educational videos at e.g., YouTube, but knowledge-centered approaches reusing available content in a standardized and generic manner (domain independent) are still missing. We are looking for mechanisms that bring more motivation and incentive to the user while interoperating with Open Content. Therefore, we chose a game-based learning approach embedding Open Content in knowledge games. Assessment methods - e.g., self-assessment during lectures - to hold students' attention integrated in Learning Management Systems (LMS) like ILIAS [27] showed good evaluation results among the students and they asked for more quiz-like interaction [28]. About 70 percent of the students did the quizzes by themselves at home again to repeat the material. Workload to prepare and realize online assessment and quizzes is very high – so we are searching for (semi)automated approaches to generate e.g., question-answer-pairs. Furthermore, the web offers a lot of information and knowledge available as Wikis, semantic or geographic data and multimedia.

F. Social Semantic Web behind the scenes

RDF: The Semantic Web, increasingly transcribed as „*Linked Data*" [29] or "*the Web of Data*", is supposed to bring new quality to the internet: What was formerly known in form of internet pages only to human beings, shall now be applied to automatic processing. In order to achieve this, the formerly in continuous text existent data will be classified, its properties transformed into defined forms and, as their aggregation, connected through labeled links. The schema of the „*Resource Description Framework*" (RDF) [12] - developed for this purpose - follows a natural speaking sentence structure. It consists out of the following information carrier: The "*subject*", "*resource*" or "*node*" is presented as an URI (Unified Resource Identifier) just like the "*predicate*" and the "*object*". All of them might contain properties following their description. Properties their self are typed and if we imagine RDF as a tree, they represent the leaves. Their type is normally declared like for example the number 42 is of the type "*integer*", but functionally dependent from its predicate. The relation of the information carriers is modeled implicitly, always directed and qualified through the predicate. Instead of speaking about "*subject*", „*predicate*" and „*object*" (the object might be a subject as well), it is more efficient to name them "*properties*" that are assigned to resources. Resources are connected in three ways over relations: As source, target and identifier.

SPARQL: With SPARQL (SPARQL Protocol and RDF Query Language) [13] a search and query language for RDF repositories is designed. SPARQL is a W3C specification

since the beginning of 2008. SPARQL's syntax is similar to SQL while columns can be defined to answer requests. Filtering expressions are possible in SPARQL that are placed in the WHERE clause in SQL for example. Actually, there is no efficient functionality to implement full text search yet. Next to ASK there are no aggregate functions available in the SPARQL specification at this time. ASK allows only a true/false statement about whether a request delivers a result or not. Abandon the special case of an identity, regular expressions should be used for full text search. Such expressions do not fit properly to a large amount of data, because up to now there are no database indices available to speed up them upon text. That is why every expression has to be evaluated for any RDF property and all of the properties have to be fully loaded too. To get more aggregate functionality next to ASK, many providers implement additionally, proprietary extensions. This up to now not standardized extensions use the strength of the traditional, relational query language SQL and a combination of SPARQL and SQL. For this reasons also qKAI does not use SPARQL only. Temporary query results have to be stored anyway, to allow acceptable performance while requesting and combining distributed RDF resources. These results are stored in a relational database – MySQL – and SQL is utilized for effective, internal data processing (see Section 4).

REST: Representational State Transfer (REST) [30] [31] is an architectural style - not tied to any particular technology - although it is used as a guide for designing architectures following four key constraints: identification of resources handled by URIs, manipulation of resources through representations using the HTTP's uniform protocol (GET, PUT, POST, DELETE), self-descriptive messages and hypermedia as the engine of application state. REST was defined by Roy Fielding [30] in his dissertation as an architectural style he used for foundational Web specifications - in particular HTTP and URIs. REST offers important architectural properties to improve reliability, scalability or simplicity. These properties are often named as superior to SOAP web services so that we can speak about REST as a "*thin*" style SOA alternative. Especially in Web 2.0 applications, REST web services are very popular these days.

Rich Clients: Regarding today's browser-based user interfaces, Rich Clients using especially asynchronous JavaScript and XML (AJAX) are a wide spread trend. The aim of Rich Internet Applications (RIA) is to bring desktop-alike and easy to use interoperation to the Web. Next to AJAX, Adobe Flash/Flex [32] and Java based User Interfaces (UI) are technical alternatives. The best technique to choose depends on the main requirements that have to be fulfilled [3]. Flash/Flex3 for example offers the advantage of less scripting work and easier event handling to reach highly interactive functionality, if the focus lies on multimedia and design issues. All these Rich Clients can be seen as an extended and enhanced view in the traditionally Model-View-Controller (MVC2) concept. Advantages Rich Clients offer are e.g., faster reaction to user requests with partially reloads of site parts without refreshing the whole site, less network traffic and server load as well as offline usage possibility. A so-called Rich UI Engine delivers the General User Interface and the presentation logic is divided from visualization components. RIAs as standalone Web clients that interact with the server side through web services are a promising combination. One of the most important advantages of the Client-Server model is the idea that the User Interface should be developed independently of the business logic and data persistence technology. Nevertheless, to be honest, in today's Web programming practice before RIA, the UI is in fact tightly coupled with the server-side technology of choice. If we like to change our backend functionality for example from Java to PHP we also have to rework all the scripts generating the HTML UI from *.jsp to *.php. To avoid this problem we can now choose a Rich Client communicating over standardized interfaces like web services only and put the most of the UI logic to client side to get a real separated solution.

II. How to utilize Open Content for higher-layered applications?

In this section, we outline what we see as requisite to turn Open Content into an organized, useful knowledge base for higher-level applications. We are aiming at the establishment of powerful, but by the user easy to handle mechanisms for acquisition and inquiry of relevant data out of heterogeneous sources. We have to serialize formats for unitary, comprehensive analysis and mediation of distributed, inconsistent content. Mediation means here to utilize input data for higher layered applications by offering personalized query plans, transformation, annotation and interoperation. Open access to knowledge and data in e.g., RDF representation brings advantages in interlinking and easily accessing distributed data on the Web. Data processing concepts allowing machine- and human-interpretable staging without storing redundant data permanently become possible by semantic interlinking.

Information chunking for easy to digest knowledge bits without losing context information is needed for better understanding and human-capable representation during interaction. In qKAI (semi-)automatic extracted information units represent a piece of information that can be qualified by annotation and interaction to a knowledge unit – always aware of its context not to lose information and to allow effective change management (actuality).

Knowledge life cycle concerns have to be matched with content cycles of the Read-Write-Web. Acquiring (inquire, discover, categorize, index) maintaining and mediating (manage, analyze, enrich, transform) and particularly reusing (interoperation for information, learning, knowledge transfer) services have to be established.

The more we know about a source, the better we can reuse it. Therefore, metadata and its annotation are essential for accurate thematic, semantic analysis and quality determination. Determining the quality of content enables us to rearrange according to source criterions like provenance, timeliness or correctness. Emerging qualitative valence of information units and sources raises information to valid knowledge. To get the emerging qKAI knowledge base applicable, interaction services for learning, rating, ranking,

inquiring, exploring and annotating are needed. Motivation and user involvement are important aspects and learning games are proficient for easily accessible, intuitive forms of interactivity. Synergy effects between learning, gaming and annotating content arise. Content enrichment by the user is seen as an implicit, positive side effect in qKAI application services.

Learning scenarios in qKAI enable self-controlled and directed concepts embedded as interaction services. We are starting with the scenarios shortly outlined in Section 6, but extension is kept simple by architectural division of presentation, its logic and managing interaction services. Learning targets deal with competency in information handling and learning in joyable manner with the Internet of Services.

RESTful SOA (Service Oriented Architecture) paradigms harmonize with Web 2.0 concerns. They support Semantic Web technologies as scalable, reusable and unified software concept, while retaining application autonomy and put the resource in the center.

III. STATE OF THE ART AND RELATED WORK

Currently some novel Social Semantic Web applications (e.g., Twine [6], Freebase [5], Knol [33]) arise that regroup existing knowledge, allow manual annotation and creation. There is still a lack in all-embracing, standardized frameworks, integration practice and reusing interoperation scenarios. A few examples for game-based interaction with Open Data like Quizzer [34] are available - embedding User Generated Content or annotating the Web of Data. Interaction and learning applications that combine arbitrary sources in an extensible SOA way are not available yet, as far as we know. DBpedia mobile [20] is an ideal example for browsable linked data combined out of different sources on mobile devices, but interactive learning scenarios, change management and further web service integration have still to be applied. SOA concepts find more and more their way into university and campus management systems. New qKAI services can be loosely coupled and integrated. Frameworks helpful for the GUI side use Asynchronous JavaScript and XML (AJAX: Prototype [35], Dojo [36], YUI [37]) or Adobe Flash/Flex [32] (e.g., FlowUI Open Source RIA Enterprise Mashup Framework). The OpenRDF Sesame framework [38] brings comprehensive JAVA functionality for semantic data discovery, querying and transformation. The Semantic Web Application Framework [39] enables further semantic tasks.

Semantic search engines like Swoogle [40], Sindice [41] or Watson [42] deal with searching, exploitation and large scale access to the Semantic Web of Data and can be used as gateway to find further relevant input resources for the qKAI hybrid data store. Nevertheless, beyond getting to know where to find these resources, qKAI wants to transform and embed the resources' content after mediation in own application scenarios. Therefore, we have to add its own open and transparent data processing, storage concept, representation and change management that is outlined in Section 4. To crawl the Web of Data we can use available solutions, but for effective storage, recombination and real time access of the resources' content during user interaction we developed the qKAI hybrid data layer. Additionally, existing crawlers are pluggable into the qKAI data layer.

IV. QKAI APPLICATION FRAMEWORK AND SYSTEM LAYER

The system design of the qKAI application framework is organized in four main layers as a combination of mediator-wrapper-concepts [10], service oriented approaches (Internet of Services) and conventional web application N-tier design. In this section, we explain the components and tasks of the applied layers as shown in Figure 2.

The presentation layer implements the General User Interfaces and its necessary logic. To fulfill extended MVC2 separation, the mediation layer presents the business logic and controller functionality. Regarded in a SOAP style, service-oriented way we would place the Enterprise Service Bus (ESB) here and the service broker belonging to the discovery manager. The mediation layer acts as middleware that connects available services (service mediation) and other technical components. The definition of "*mediation*" in qKAI is also interpreted according to Wiederhold [10]: „*A mediator is a software module that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of applications.*"

The data layer meets the model level in the Model View Controller pattern and extends it with wrapper services at the

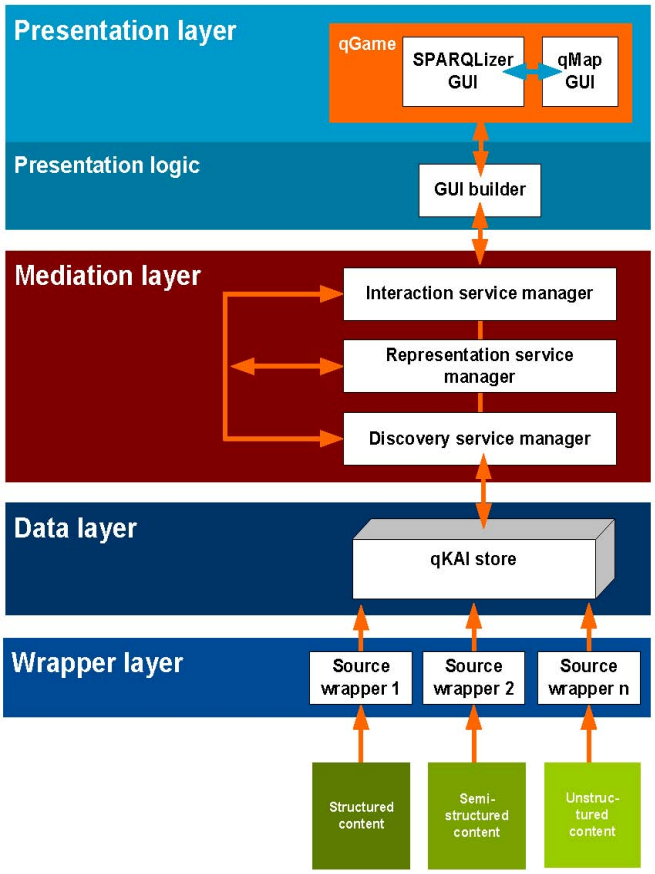

Figure 2. qKAI application framework: System layers as conceptual design and development basis

wrapper layer to embed various distributed sources. The data layer has to manage hybrid data processing enabling RDF and XML related data as well as relational database content. Existing data sources are temporarily retained for mediation purpose. qKAI processes provide new generated knowledge as open RDF or serialized JSON representation after mediation.

V. THE QKAI HYBRID DATA LAYER: SOME DETAILS

The qKAI knowledge representation consists for example of RDF graphs and superior metadata about them. Existing graphs outside of qKAI are first stored as link to the origin source in the qKAI service and source repository e.g., using the URI of a SPARQL endpoint like http://DBpedia.org/sparql. New generated information is stored separately and sustainable at the qKAI data layer. The data processing concept contains a persistent, relational database component, flexible representation (RDF, N3, JSON) and temporary fetching of working data during discovery and enrichment. To allow persistent storage and to handle relational data with only a small set of classes in an effective way we deployed the Java persistence API (JPA) with the EclipseLink implementation [43] and Builder patterns.

Linked Data concepts [29] enable persistent, resource-related storage, change management and non redundant, context-aware data processing by interlinking identifiable distributed resources. By qKAI services generated knowledge about available sources is stored additionally in MySQL and can be represented as Linked Data on demand. Figure 3 illustrates, how the qKAI data layer embeds Linked Open Data resources and represents its data store as another node in the Linked Open Data cloud. Following the Linked Data keynote and overall desire of next generation web applications, to combine different, distributed RDF stores on demand in real time without buffering, our first trials to embed content while querying exemplary the DBpedia endpoint, failed. Without fetching a copy of relevant data, results showed that bad response times that we had to develop a more practically solution. Main reason for the bad response times is the SPARQL endpoints performance as outlined in the following (see Section *"Fetching and indexing"*). Now the qKAI data layer buffers relevant resources in a traditional, relational database structure to allow adequate performance to users' requests and ability to further processing and annotation of the acquired content. Further on, affordable hardware can be used to reach good performance results without the need to investigate in high-end enterprise server technology.

This section gives an introduction into the qKAI data management based on the concept that every RDF resource is portable into a relational database structure and vice versa. DBpedia is used as an exemplary knowledge base accessed via its SPARQL endpoint to demonstrate first results while building the hybrid qKAI data store out of distributed web resources. Main criteria for the qKAI data store are easy, affordable, scalable reusability, possibility to mediation and representation of acquired resources of different kinds and varying provenance. Points of Interest (POI) are deployed to give a starting point into the knowledge base by users or further applications and to update the qKAI store partially on demand in an effective way.

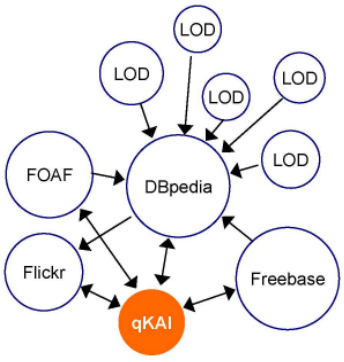

Figure 3. qKAI and the LOD cloud

A. *Discovering Linked Data by setting Points of Interest*

Because of the wide web discovery space (open world assumption), conceivable performance problems answering, and updating comprehensive queries in real-time, among others we are integrating POI (Point of Interest) setting functionality in qKAI. Further applications and the users need an interface to define points of interest and to limit inquiry space and knowledge base according to own interests. Setting and storing several POIs according to different domains and themes becomes possible. Once set, POIs are stored for reuse and are interlinked with each other. The POIs enable defined entry points to the large RDF knowledge base and enable huge databases to update temporary redundant stored data to be efficient synchronized with their provenance source on demand when requested. Interesting parts of domains and knowledge bases will be updated even often then irrelevant ones and weighting is set this way implicitly for further statistically tasks. We implemented a thin, asynchronous updatable and practicable data layer this way. Browsing Linked Data stores requires semantic browsing functionality for different devices like web browsers or mobile devices. Representational Web services are designed to fulfill device specific representation requests of the qKAI knowledge base.

A single geographically and/or thematically oriented Point of Interest (POI) is now using the qKAI implementation completely reproducible by the qKAI data layer through a single and effective SQL query. In the current implementation, it is assumed that the topic the user is interested in is a part of the knowledge base yet – now DBpedia for testing purposes. DBpedia contains all thematically categories out of Wikipedia – so we know approximately, what we got up to now. Under this condition a thematically search space limitation is representable through the following SQL statement:

```
-- POI for persons
SELECT DISTINCT r.resource, r.resourceURI
FROM resources r
JOIN relations l ON (r.resource=l.resource)
WHERE l.predicate
= ANY (SELECT resource FROM resources WHERE
resourceURI="…/22-rdf-syntax-ns#type")
AND l.target = ANY (SELECT resource FROM
resources
```

```
WHERE
resourceURI="http://DBpedia.org/ontology/Person");
```

POIs can be combined without limitation. In SQL, this means an implementation as follows („NATURAL JOIN“ is a substitute for „INTERSECT“ which is not supported by MySQL):

```
SELECT resource FROM
(/* SELECT from POIa */)
NATURAL JOIN
(/* SELECT from POIb */)
NATURAL JOIN … -- for further POI
```

The derivation of this high-performance, multithreading solution to embed distributed Linked Open Data in higher-layered web applications is exemplary enlightened in the following. A **minimalistic data model** with **maximized depth** is applied, because the implementation is suitable for several semantic data purposes on a Quadcore hardware platform that is available for about 500 € (May 2009). Because every table can be mapped to RDF and vice versa, there is no need for further extension. Further on, relational data storage brings the advantage of flexible, independent representation and delivery on demand e.g., in JSON next to RDF/XML or N3 Turtle.

B. SQL replaces SPARQL

Considering the current infirmity of SPARQL shown in the introduction and regarding the fact that popular SPARQL implementations are still a facade for relational databases, it seems to be consequent at this time to abstain from the broad adoption of SPARQL. Significantly, even DBpedia uses Virtuoso for a SPARQL endpoint with proprietary extensions, but MySQL is used for query processing in the background. Thus, we decided to choose MySQL directly for the qKAI data layer and reduced the data processing bulk to SQL that most developers are more familiar with up to now. On the one side, we can resort to a broad pool of proofed SQL solutions – enabling hierarchical queries for example – on the other side following developers save incorporation time to get started with the qKAI data layer.

SPARQL is used only to acquire DBpedia content or other RDF stores using hybrid indexing. Our first approach, sending complex SPARQL requests without SQL background support to the endpoint, failed. Even for a tree out of more than two nodes, results often could not be returned and the endpoint timed out. Under these conditions, we discarded this approach. Present practice showed that too many dependencies on distributed resources constrain further work that relies on the qKAI data layer, because the reliability of reaching SPARQL endpoints is not well calculable this times.

C. Hybrid knowledge index

An initial qKAI knowledge base e.g., out of DBpedia can be easily imported using the qKAI dump reader. This initial knowledge base is complemented on demand by the proper connected resources via SPARQL. The dump reader accepts requests and queries the buffered qKAI data layer instead of the distant resource. A hybrid knowledge index arises this way by and by.

D. Change management

The in MySQL buffered qKAI knowledge base will become obsolete earlier or later. Thereby the qKAI data store might be soon as large that synchronizing all entries with their provenance resource is not possible anymore all at once. The hybrid knowledge index allows updating the buffered data store partially at runtime. If a user or an application signalizes that more elements out of e.g., DBpedia are needed while setting Points of Interest, they can be additionally loaded and their content can be updated on demand. If the update amount and interval is set properly, “*fluent*” actualization of the whole qKAI knowledge base is possible. Only small and relevant amounts of data have to be fetched out of the provenance source this way and they can be parallel processed in the qKAI data layer. Data can be provided reliable at any time this way.

E. Reusability and extensibility

All database connectivity of the qKAI data layer is implemented autonomic and persistent without relying on other qKAI components. To embed further SPARQL endpoints we only have to deploy the new endpoint’s address. The implementation of generic classes allows using even different protocols – what we showed exemplary with the qKAI dump reader. New sets of POIS can be integrated the same easy way. At this time, we offer POI setter for thematically, geographically and full text POIs. Full text POIs search like traditional search engines for matching phrases.

F. Multithreading

Without to beware of it, most developers of web applications work with multithreading on multicores. In case of Java, the Container manager treats every request as a separate thread. Most of the time there are no negative side effects, if the result is not especially tweaked for this purpose. A different case we got with the qKAI data layer that is forced to be driven parallel to reach practicably performance. In addition, blockades while querying to slow answering resources had to be avoided.

G. Fetching and indexing

A first, not yet thread optimized version of the qKAI data layer claimed over 30 seconds to fetch some test data and all of their properties. Most of the time was wasted with the one thread of the application waiting for DBpedia answers.

To understand the underlying processes and their problems to infer a more reactive solution, we analyzed the tasks to do while fetching our test data from a SPARQL endpoint:

1. For a given word the URI has to be found.
2. All properties and relations of the constant URI and their neighborhood are fetched. This happens

gradually because SPARQL does not allow binding constants to variables.

3. Step 2 has to be repeated for every fund.

H. DBpedia Search example with the term „Nietzsche“

Now we take the famous search term „*Nietzsche*“ , looking it up in DBpedia using SPARQL only and we get the following results:

TABLE I. NECESSARY REQUESTS SEARCHING FOR “NIETZSCHE” IN DBPEDIA

Step	Document count	Requests
1	444 hits for search term „Nietzsche“	1
2	8458 properties 41804 outgoing links 2068 mutual properties among	444

Table 1 shows the necessary steps, document count and requests searching for the example term “Nietzsche” in DBpedia using SPARQL only. In general, all hits have to be loaded, because SPARQL does not know any word quantifier ordering and the wanted term might be at the end of the result list. We get a serial waiting time for 445 query results with 50.000 new created documents or even markers. This result can be constructed much shorter enabling qKAI buffering DBpedia and then applying the following SQL queries to the qKAI data layer:

```
SELECT resource
FROM properties_fulltext
WHERE q="Nietzsche"
```

Repeated for table „properties“:

```
SELECT COUNT(*)
FROM (
SELECT resource FROM properties_fulltext
WHERE q="Nietzsche"
) sub JOIN relations p ON (sub.resource=p.resource);
```

Also for “relations”:

```
SELECT p.property, COUNT(*) amount
FROM (
SELECT resource
FROM properties_fulltext
WHERE q="Nietzsche"
) sub JOIN properties p ON (sub.resource=p.resource)
GROUP BY p.property HAVING amount > 1;
```

We noticed that a trivial DBpedia request is answered in 2 seconds at its best. Thus, without synchronous loading we have to wait 14 minutes at least. To solve this problem, we divided the process in a pipeline of five steps to identify especially working and waiting intensive tasks. Finally, we matched these tasks to a suitable number of threads and processor cores. In our implementation, the Java classes do not communicate directly. Instead, they use the „*FetchingMediator*” as broker. For every step represented through a class, a predefined amount of „*ThreadPoolExecutors*“ is started. Class instances are lined in a queue. Every started thread will work constantly and multiple threads accomplish tasks parallelized. With four concurrent „*RelatedResourceProducern*“ the waiting time of the above mentioned example theoretically increases to a quarter. Practically DBpedia answers slower with increasing concurrent number of request. Next to qKAI there are several other applications querying DBpedia what makes the answering times unforeseeable. Waiting times of several minutes cannot be foreclosed.

The **parallelized approach** prevents the bottleneck in qKAI: Following requests might be answered faster allowing queued search terms to be processed further on despite of small blockades.

I. Evaluation of the qKAI hybrid data layer

Initial search space creation without preselected data dumps: If data is fetched only by requesting a SPARQL endpoint like DBpedia without buffering, the used time until the whole search space is available, depends on its connection. qKAI itself uses only very simple SPARQL expressions and - if implemented in the endpoint - the Virtuoso extension ”*bif:contains*” [17] to allow faster full text search. According to our samples it takes about 20 seconds until 10 results are available: To be presentable, a result has to be found and must be fully loaded as well. Resulting out of this unacceptable response times while only using SPARQL requests on distant resources, qKAI uses a prepared data store as copy of the provenance resource to utilize distributed data for further application scenarios in real time. Therefore we currently dissuade from deploying “*SPARQL only*” installations in productive environments – even if this approach is desirable regarding the long term, it is not realizable in an affordable way up to now. Instead, a minimal data store - with at best domain or application customized data - should be prepared like explained in the following using the example of the qKAI data layer.

Initial search space creation with preselected data dumps: A “*dump*” is a view of a database written down in a file. In case of DBpedia, we speak of 40 compressed files that reach decompressed 25 GB. The decompressing time took about half an hour on test system 2 (see Table 3). This data has to be transformed from N-Triples format into a relational data base structure to allow more effective processing and to optimize further queries. N-Triples are a rather redundant data structure because e.g., a RDF subject is repeated for each of its properties. Directly transferring of N-Triples into the qKAI data store is a quite ineffective task. It would mean to get tables with over 200 million lines and expensive JOINs with more than one property concurrently. The up to now fastest known SPARQL implementation Virtuoso [17] uses just like qKAI OpenRDF Sesame [38] to parse data into a more effective representation. After cleaning up, the data are transformed into the qKAI data model. Therefore we turned off the database indices in MySQL because it is a rather time intensive task too.

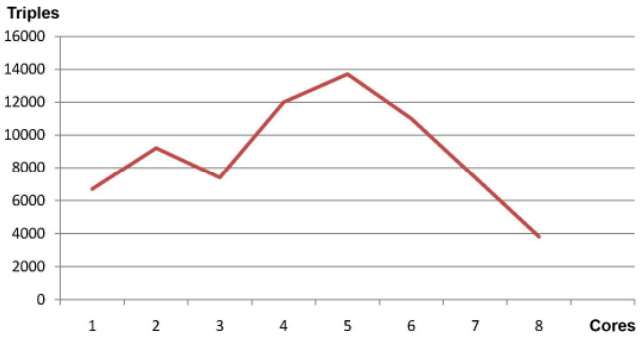

Figure 4. Count of read triples per second and amount of processors per second, measured over 80h, while reading in DBpedia dumps.

Figure 4 visualizes reading in the DBpedia dumps on test system 2 (see Table 3). The peak lies at five by qKAI used processors because there is an optimal balance reached between 5 cores for qKAI resources and 3 cores for MySQL. In average, even 4 cores fit well.

J. *Result concerning the qKAI hybrid data layer*

If a distant RDF store like DBpedia is mapped to the qKAI data layer as relational structure, efficient search space limitation and processing of the content is realizable with good performance using internal SQL selection. Requesting and combining distributed RDF resources using SPARQL only currently turned out to be not practicable. Therefore, we implemented the transformation into an own, efficient relational data structure. At this time, the qKAI data store consists out of DBpedia. We will enhance it with further resources by and by. The qKAI data layer can be used to embed available semantic search services to build a hybrid knowledge index for the Semantic Web - and in future work for further Open Content resources.

Embedding e.g., the Sindice and Watson crawling functionality is under development. Sindice offers a large RDF data index; qKAI is able to mediate with to acquire further resources next to DBpedia. The algorithm for change management introduced in Section 4 can be used for periodic and on demand synchronization with provenance sources. Relying on a modular architecture, qKAI can be easily extended by further Points of Interest e.g., for multimedia content.

K. *Test systems*

The implementation of the qKAI data layer is tested on two different systems to determine the ideal distribution strategy:

TABLE II. BENCHMARK DATA FOR TEST SYSTEM 1

Processor	AMD Phenom 9550, Quadcore, 4x 2,2 GHz
RAM	8 GB DDR2 667MHz RAM
Internet connection	14 MBit
Storage	Four Western Digital WD6400AAKS SATA 2 hard disks, Adaptec 3805 RAID Controller in RAID 5
Throughput storage	380 MB/s read, 300 MB/s write

TABLE III. BENCHMARK DATA FOR TEST SYSTEM 2 (AMAZON ELASTIC COMPUTING CLOUD 2)

Processor	2x Intel XEON, Quadcore, 8x 2,33 MHz
RAM	7 GB, unknown
Internet connection	1 GBit
Storage	Amazon EBS Storage, Western Digital WD6400AAHS
Throughput storage	60 MB/s read, 50 MB/s write

VI. SERVICES AND COMPONENTS IN QKAI

To keep the qKAI application structure flexible, extensible and autonomic, we decided to encapsulate functional subtasks in small web services. Web service interaction will follow RESTful Web 2.0 paradigms. Self-descriptive messaging is the most important constraint of REST. It means that every message needs to include all the information necessary in order to understand the message itself. We do not see a web service description language like WSDL as mandatory for REST – also it is possible to describe RESTful web services using WSDL 2. The fundamental advance of REST over the styles of e.g., SOAP or CORBA is that all service interfaces are the same. There are no differences with the need for explicit description. After registering all available services and resources in the qKAI data store according to a broker, we are looking towards embedding structured databases by converting them to RDF. The last challenge is to enhance unstructured content for qKAI integration developing more complex wrapper services.

We are dividing qKAI functionality into two main developer levels:

1. RESTful backend web services for acquiring, selecting and representing textual and multimedia data. Read and write access to resources is performed over the HTTP protocol by the GET and POST method. Working at this level means extensive use of the Java Jersey API (JAX-RS specification) to build atomic web services for several subtasks in effective manner.
2. Rich User Interface components let the user interact with different kinds of activities. Frontend components are built with AJAX and/or Flash/Flex according to their type of interactivity.

A. *RESTful knowledge engineering*

We deploy RESTful web services (atomic and composite) to handle standard tasks: acquire, represent, transform and annotate resources. Loosely coupling of remote resources and services becomes possible; stateless services and server communicate over the http protocol.

B. Modular rich frontend

According to Rich Internet Application (RIA) concepts, user interfaces are executed at a stateful, client side Flash runtime. Desktop-alike applications offer advantages like faster reaction to user requests, less network traffic, less server load and offline usage possibility. We decided for a Rich Thin Client using Adobe Flash and Flex - so business logic remains at server side. The Flash player acts as rich UI engine and delivers the GUI. The presentation logic is divided from visualization components. Flash is opening up its format with the Open Screen Project [44]. The Flash format and Flex SDK are not that proprietary anymore like some years ago. Nowadays the Flash plug-in is spread over 90 percent around browsers. Flash also is promoted through spreading its new FLV format in online video communities like YouTube. So Flash obtained enormous ubiquity and lost a lot of its proprietary nature these days. Browser, platform and device independent developing interactivity is the big advantage of Flash/Flex applications compared to Ajax. Flash is predestinated to design gaming content because of its effective possibilities with high design issues and focus on usability.

C. Interaction and game-based activity

As Ahn's reCAPTCHA, ESP game [22] or Amazon Mechanical Turk established gaming as a well-suited instrument to solve several tasks in knowledge engineering. However, they do not mention any learning or knowledge concerns while gaming. On the one hand, users can enrich content; on the other hand, users can learn and share knowledge through gaming with Open Content in a social incentive and challenging way. We are aggregating existing information and enriching it while interacting with Open Content. Even statements about contents' quality can be deduced out of users' content and activity. Especially fact-related knowledge can be transferred and learned, if resources are presented in rule-based manner to the user and he has to solve predefined learning tasks earning rewards.

Creating and editing: We support authors to create and combine content. For example, false answers are automatically generated by wrong-answerizer services, if a question is generated by a user. qKAI offers proposals for new learning games out of given domain knowledge and concepts. Matching between available content and suitable gaming types is described in ontology-based concepts. At the same time, the user enhances underlying ontologies while deploying game types and rating gaming content. We want to create self-organized ontologies that adaptively grow with ongoing user interaction.

Interlinking and grouping: Grouping of existing resources and interlinking with missing ones is rewarded e.g., with in-game incentives.

Rating and ranking: Instead of simple questionnaires, rewarding concepts are deployed to get user feedback.

D. Educational aspects and suitable domains

Gaming is not overall suitable to learn every skill in any domain. Some educational tasks or topics are learnable and transferrable more effectively utilizing certain gaming types then others. Furthermore, content has to be divided into different difficulty levels and tasks for distinct audiences. Our game-based learning concept is not limited to a certain audience. For example, undergraduates can learn with gaming content of lower difficulty level and other topics then students in higher education. Game creation is a gaming and learning challenge by itself – so lecturers can choose suitable content out of the gaming pool and add own missing material or further web resources, where necessary. We identified the following domains so far as most suitable to embed for further evaluation purposes: Geography, architecture, history, events, persons, medicine and health. Overall, every domain seems to be suitable for our social educational gaming approach, if learning aims can be fulfilled while creating, answering and querying factual knowledge and predefined learning tasks (especially recall and rearranging of factual content). Popular examples are multiple-choice questions, text-text assignment, image-text assignment or ordering questions. These question types have the advantage, that they are also available as learning standards in Learning Management Systems. They can be easily converted into IMS/QTI after in-game creation. Embedding multimedia like zoom parts out of images or video/audio sequences is also possible. Next to knowledge unit gaming types, we are implementing and researching location-based gaming types relying on geocoded information and correct geographically placement. Here, we can visualize information in a very handsome and effective manner using Yahoo! Maps and their Flash/Flex API to interact on map-based interfaces.

E. Further techniques and libraries

To perform backend and frontend tasks, there are some available and proofed libraries merged up and extended. We are developing qKAI as a Java web application with Servlets and Java Server Pages deployed in a Tomcat Servlet Container [45]. The frontend is next to AJAX designed using Adobe Flex and Flash components for highly interactive tasks with extended design issues (e.g., learning game sequences). Most of the qKAI service and component offer is reusable in nearly any information and knowledge transfer scenario. Additionally some gaming services and components allow specialized functionality for social educational gaming.

F. Project management and build automation

qKAI is using the build and management tool Apache Maven [46] with support of Subversion to automate working tasks. qKAI offers fast start up and good documentation facility of all working processes to reuse them as much as possible this way. Interface classes can be used without knowing about the classes' Weaving. In addition, the standard for (JUnit) tests has been adopted with deploying Maven. The programmers get hints about code snippets where better handling of the framework and its language is suggested. Maven dissolves all dependencies among packages and downloads them automatically in the right version. Developers can use their IDE of choice. Adjusting qKAI to e.g., Netbeans or Eclipse is done through one

Maven command. The server and the database qKAI are currently deployed on, can be changed very easy, too.

VII. FURTHER QKAI USE CASES AND APPLICATION SCENARIOS

qKAI services enable embedding, querying and enriching distributed web sources for any kind of higher-level application that likes to integrate a broad, structured knowledge base with interoperation ability based on a suitable tools and services collection. We are focusing on process plans suitable for learning - realized by composite services. In the following, we give a few precise examples for use cases that are currently under development using a first prototype of the qKAI framework.

qKAI will publish services in a service broker way for reuse in other web applications like Learning or Content Management Systems.

A qKAI user for instance places a question like "*Which authors are influenced by Stanislav Lem?*" using the SPARQLizer, gets five automatically generated Multiple-Choice answers presented with relational author images from Flickr [47] and he has to choose the right ones out of it. Currently we are developing question-answer-learning-game-types generated out of well suitable Open Data in RDF format. We see two main modes as most interesting for creating first learning game scenarios out of Open Data: Players assign questions to (semi-)automatically extracted information units. Players create questions and get automated answers, which are transformed, into playable answers.

A. qMATCH

This is a prototype of an image-term assignment gaming type. First, the user enters a term he likes to get images about. Then qMATCH presents randomized terms and images out of Flickr and the player has to assign the right term to the right image via Drag & Drop assignment. Here we need a service called wrong-answerizer to assign wrong, but not stupid answers. Wrong-answerizer is deployed in further gaming types. qMATCH is useful to enhance e.g.,, language skills, geographically, architectural or historical knowledge. A conceptual gaming sequence is visualized in Figure 5.

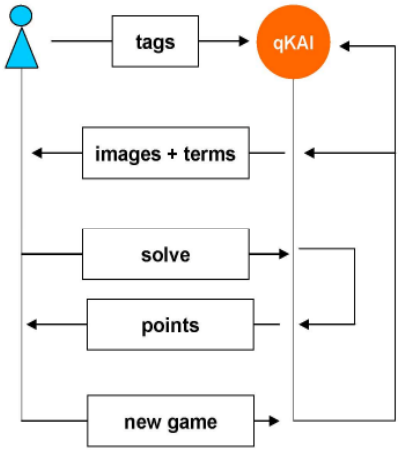

Figure 5. qMATCH gaming sequence

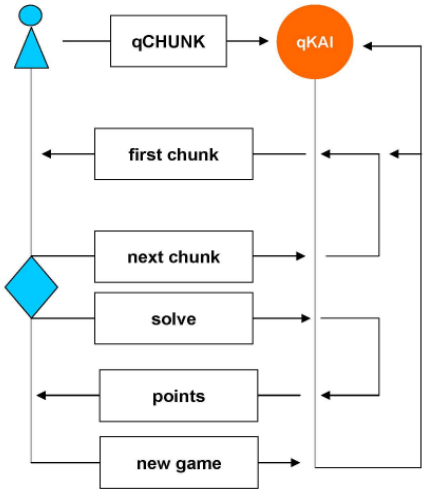

Figure 6. qCHUNK gaming sequence

Figure 5. If we use term-term assignment, a lot of vocabulary out of various domains can be assessed: assigning English to German translations, assigning buildings to right historical epochs or assigning cities to the right countries.

B. qCHUNK

This is a prototype for a text chunk guessing game based on e.g. Wikipedia articles. qCHUNK presents small textual chunks and the player has to guess the quested term with as less chunks as possible. A conceptual gaming sequence is visualized in Figure 6. Multimedia chunks like zoom parts out of images are conceivable too. The chunks are extracted deploying the SentenceDetector of OpenNLP [18]. The guessable term is exchanged with a placeholder like "?" and is displayed to the user. The user gets 20 seconds to solve or to switch to the next chunk related to the guessable term. qCHUNK is also suitable to integrate multimedia e.g.,, images, sounds or videos.

Example:

- Chunk: ?? is the capital of Lower Saxony founded in 1942.
- Answer: Hanover.
- Next chunk: ?? is located at the Leine.
- Next chunk: image of Hanover city.

Often chunks imply indirect questions like "*Is ?? the capital of Lower Saxony*". The user will get the ability to create a question out of the content that is displayed. While storing and processing hypermedia, it is mandatory that chunks do not lose their origin context and provenance.

C. qMAP

With qMAP a map-based geocoding frontend is under development. Questions, answers and their combination (knowledge units) will be placed and enriched with geocodes at the qMAP. qMAP acts as a kind of gaming board for placing, exploring and editing stored question-answer-pairs. The qMAP will interact with SPARQLizer (see Section D) and represents submitted questions and related answers.

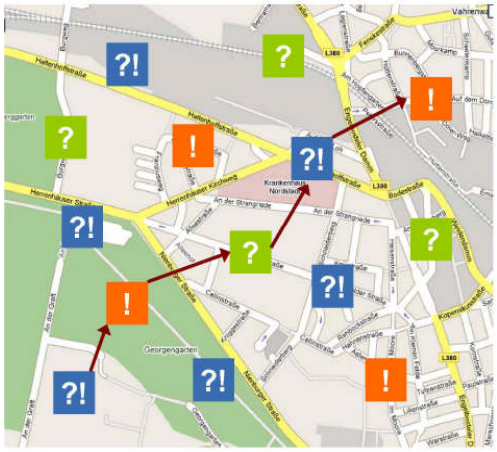

Figure 7. qMAP concept

qMAP offers placement of locations, events, buildings, photos or persons. We provide interaction like filtering, searching, editing and adding knowledge. OpenStreetMap [48] or Yahoo! Maps [49] are good alternatives to Google Maps [50]. Map symbols are connected with different gaming interactions and information units.

D. SPARQLizer

With the SPARQLizer a visual query interface is under design, that allows intuitive question to query plan transformation on distributed RDF stores deploying SPARQL endpoints and dynamic graph interlinking. User-generated SPARQL queries are stored as graphs enlarging the qKAI knowledge base and ready to query against in further query plans.

E. Annotating and qualifying services

Joker option services for annotating, rating and qualifying are currently under development belonging to the interaction manager of the mediation layer. qKAI jokers allow game-based functionality to add additional sources and to qualify metainformation by rating and ranking input to the qKAI knowledge base. Playing the "*Know-it-all-Joker*" bounds the player to add a source (or information) that proves contrary statements. The "*Nonsense-Joker*" marks an information unit as semantically wrong or inconsistent and defers it to review mode by other qKAI users. The "*Hint-Joker*" allows looking up related sources or other users' answers as solution suggestion. The "*Explorer-Joker*" allows exploring the right answer on the web outside of qKAI during a predefined time. The "*History-Joker*" enables lookups in played answers, ratings of other users by logged interaction and transaction protocols. Statistical protocol analysis is suitable to infer further metainformation.

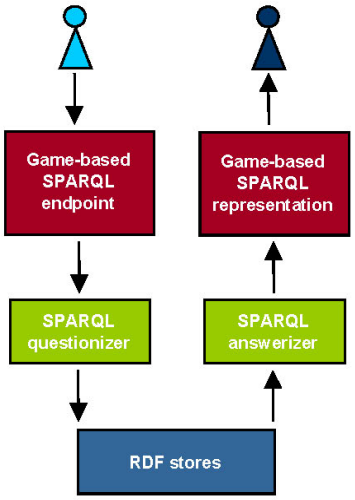

Figure 8. SPARQLizer concept

F. Social knowledge profiles

Personal profiles and self-reputation are a popular, ongoing web trend. There are useful Single-Sign-On solutions like OpenID [51] or FOAF [52] files to avoid redundant profile storage in different web communities. Knowledge and learning related properties are enlightened less. qFOAF aims at building personalized, transparent knowledge profiles to connect users additionally thematically and semantically. qFOAF profiles are visible to other users and list a statistical overview of the own knowledge activity. Categories applied contain domain interest and expert level with detailed scores and ranking. This profile serves allied and alienated players as a hint for further activity as known in common games. qFOAF builds a qKAI resource in RDF at the beginning of a game as extended FOAF file with unique URI for every user. It connects the user with topics, knowledge units or other players systematically while gaming. The qFOAF file can be enriched (semi)automated with given information by the user. Existing FOAF files, geocodes or interests, can be included while gaming and interacting with game points and content. Examples for gaming content are created questions, answers, knowledge units, ratings, resources, domains, locations or friends.

G. Global point and level system

A global point system is provided to document learning progress, personal interests and to implement incentive and reputation ability. Every kind of interaction is rewarded with qPoints according to its grade of interoperation.

Incentive for user participation is implemented as globally rewarding system of any interaction (qPOINT, qRANK). "*Knowledge is power*" is the simple conceptual slogan on top of qKAI gaming. The more users interact and adapt knowledge, the more they will be rewarded. Adapting knowledge is possible while solving learning tasks on your own or in alliances. Single player and alliances can challenge with each other offering knowledge battles of certain topics. A look into qFOAF user or alliance profiles allows to estimate the own chance to win a challenge or battle. We will enable to steal knowledge from others. Knowledge will become conquerable this way. Alliances and single player will be able to own others knowledge by solving game tasks. An underlying point based rewarding system can be converted into avatar items and further awards. In the future, gaming with qualified content might bring gaming points

into reality by allocating them with test examinations to offer further incentive to students. The global point and level system documents learning progress and personal knowledge. qRANK services allow game-based rating and ranking of resources by levels and in-game joker options. After users have played jokers, ranking, annotation and even semantic correction becomes possible.

VIII. CONCLUSION AND FUTURE WORK

We introduced the qKAI application framework for utilizing arbitrary Open Data sources and services in a standardized, RESTful manner aiming at highly interactive scenarios in information and knowledge transfer. To keep qKAI easy extensible with reusable, autonomous service design, we added next to a slim, RESTful SOA a 4-layered mediator-wrapper-schema to the system specification. qKAI combines Web 2.0, Semantic Web and SOA paradigms to apply and enable the Internet of Services for higher level, user-oriented applications with open access.

We are implementing first educational gaming prototypes and we are gaining at further features to utilize current web trends like social interaction for learning purpose and sustainable, knowledge-related interoperation on Open Content. Current focus lies on implementing use cases like the SPARQLizer and qMAP based on the qKAI data layer. SPARQLizer needs a dynamic, adaptive GUI, template and ontology structure to support users in (semi)automated question and answering generation out of SPARQL requests. Chunking information into capable knowledge units is work in progress. The specification of qKAI information units is done and exemplary implemented based on live Wikipedia content. Requirements are derived to utilize Open Content for social educational gaming and standard tasks in knowledge engineering. Atomic services and frontend components allow rearrangement and are suitable for any other purpose in information and knowledge transfer – not only for game-based learning. Services are adaptable to several domains.

Reusing and composing is a precept at all levels in qKAI, the challenge is to merge and to expand: Resources, ontologies, web services or available frameworks. We are aiming at standardized, machine- and human readable staging of Open Content with lightweight interoperation to develop incentive, web-based knowledge transfer and learning scenarios. Therefore, we deploy Linked Data, Rich Clients and REST. The qKAI application framework serves as conceptual basis and system specification for further work exemplary highlighted in this contribution. We want to offer learning scenarios based on user-oriented web services with lightweight interaction grounded on Open Content. Therefore, we have to implement standard tasks of knowledge engineering for extended interaction as a generic application framework. qKAI combines and respectively extends available Java APIs for subtasks to a scalable, reusable and unifying software concept.

Overall, our research focus lies on three main aspects:

- Provide standard tasks of knowledge engineering (acquisition, formalization, representation, visualization).
- Determine and enhance quality of content (analyzes and enrichment of metainformation. User's opinion and knowledge serves to annotate, rate and rank content).
- Tackle extended interaction and incentive for user's attendance while interoperating with Open Content.

We implemented an aware compact and hybrid data layer that serves as basis for further knowledge-processing applications and as interface to semantic search engines and RDF stores. The up to now missing possibilities of the therefore designed query language SPARQL are substituted by the transformation into a relational database schema. Resulting models are flexible and reusable, because any relational data structure can be mapped onto. The chosen approach of parallel data fetching and processing was not that easy to implement concerning thread safe programming, but it is very profitable regarding sustainable deployment - independent of external, at this time still unsecure circumstances:

- Time intensive requests will not block following queries in qKAI.
- Only this way data can be loaded additionally with adequate response times, if a request is not answerable through the pre-buffered data store.

Our research showed that up to now the power of SPARQL is not yet applicable to the efforts of the qKAI data layer. Even if suggested extensions like OpenLink [17] are integrated into the final specification, it is not obvious up to now, whether SPARQL alone will be properly suited for large, distributed data requests soon. Currently we noticed an ongoing trend to deploy relational database concepts because of their majority and performance – like nearly all SPARQL endpoint and RDF repository implementations do [53]. Future work in qKAI is to implement available algorithms and APIs to crawl distributed resources and to amplify the optimization of the qKAI data layer. Some MySQL optimizations like the Sphinx full text indexer [54] are implemented yet to allow faster search. For geographically search space, a GIS extension is ready to use for periphery searches.

All over, the interface to Open Data resources over SPARQL endpoints offers new ways to combine, enrich and explore the data web as an open, semantically connected graph. Data sources are presented in a standardized, machine interpretable and extensible manner deploying RDF repositories. They offer many advantages like open Linked Data ability and annotation compared to traditional relational databases under closed world assumption. Further SPARQL functionality is under development and will be part of its specification soon. We decided to mix up proven and new concepts to get a nowadays practically data layer implementation while buffering preselected data dumps in a relational database and acquiring Open Content out of RDF

repositories using SPARQL. Additionally we developed with POI setting for applications and users a practically solution to update pre buffered resources partially on demand when they are requested.

ACKNOWLEDGMENT

The BMBF (Bundesministerium für Bildung und Forschung) co-funded this work under the HELCA (Hannover E-Learning Campus) project. We thank especially Mark Kubacki and Jan Hein for their support in implementing and advancing the qKAI application framework.

REFERENCES

[1] M. Steinberg, J. Brehm, “Towards utilizing Open Data for interactive knowledge transfer”. Proc. DigitalWorld 2009, International Conference on Mobile, Hybrid, and On-line Learning (IARIA’s eLmL), IEEE Press, 2009, pp.61-66, doi:10.1109/ eLmL.2009.13.

[2] T. Berners-Lee, “Web 2.0 and Semantic Web”, http://www.w3.org/2006/Talks/1108-swui-tbl/#(1), W3C, visited 10.04.2009.

[3] M. Domenig, “Rich Internet Applications und AJAX”, Entwickler-Magazin, http://www.canoo.com/news/entwickler.pdf, 2006, visited 15.05.2009.

[4] DBpedia, http://dbpedia.org/About, visited 20.05.2009.

[5] Freebase, http://www.freebase.com, visited 25.05.2009.

[6] Twine, http://www.twine.com/, visited 27.05.2009.

[7] H. Mandl, G. Reinmann-Rothmeier, “Unterrichten und Lernumgebungen gestalten”, Forschungsbericht Nr. 60, Instituts für Pädagogische Psychologie und Empirische Pädagogik, München, Ludwig-Maximilians-Universität, 1995.

[8] IMS/QTI, http://www.imsglobal.org/question/, IMS Global Learning Consortium, Inc., visited 25.05.2009.

[9] C. Schroth, T. Janner, “Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services”, IT Professional, Volume 9, Issue 3, 2007, pp 36 – 41.

[10] G. Wiederhold, “Mediators in the Architecture of Future Information Systems, IEEE Computer, Journal, 25(3), 38-49, 1992.

[11] Open Knowledge Foundation, The Open Knowledge Definition, http://opendefinition.org/, visited 25.05.2009.

[12] RDF, http://www.w3.org/RDF/, W3C, last update: 2008.

[13] SPARQL, http://www.w3.org/TR/rdf-sparql-query/, W3C, visited 25.05.2009.

[14] J. H. Dieter Fensel, H. Lieberman, and W. Wahlster, “Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential”, The Mit Press, Massachusetts, 2005.

[15] C. Bizer, R. Cyganiak, “D2R server: Publishing Relational Databases on the Semantic Web”, http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/, visited 27.05.2009.

[16] Aperture, http://aperture.sourceforge.net/, Aduna, DFKI, visited 27.05.2009.

[17] Openlink Virtuoso, http://virtuoso.openlinksw.com/, visited 27.05.2009.

[18] OpenNLP, http://opennlp.sourceforge.net/, visited 27.05.2009.

[19] F. Naumann, “Quality-Driven Query Answering for Integrated Information Systems”, Lecture Notes in Computer Science Vol. 2261, Springer, 2002.

[20] C. Becker, C. Bizer, “DBpedia Mobile: A Location Enabled Linked Data Browser”, 17th International World Wide Web Conference WWW2008, China, 2008.

[21] T. Heath, E. Motta, “Revyu.com: a Reviewing and Rating Site for the Web of Data”, Proc. ISWC 2007, International Semantic Web Conference, Lecture Notes in Computer Science 4825 Springer 2007, pp. 895-902.

[22] L. Ahn, S. Ginosar, M. Kedia, R. Liu, and Manuel Blum, “Improving Accessibility of the Web with a Computer Game”, International conference for human-computer interaction, CHI 2006, Canada.

[23] GuessTheGoogle, http://grant.robinson.name/projects/guess-the-google/, visited 27.05.2009.

[24] Scoyo, www.scoyo.de, visited 27.05.2009.

[25] Playfish, www.playfish.com, Megazebra, www.megazebra.com, Zynga, www.zynga.com, visited 27.05.2009.

[26] Facebook, www.facebook.com, MySpace, www.myspace.com, visited 27.05.2009.

[27] ILIAS, http://www.ilias.de, visited 27.05.2009.

[28] J. Brehm, M. Steinberg, "Eine Arbeitsumgebung für elektronisch unterstützte interaktive Lehre", 36. GI-Jahrestagung 2006 - Informatik für Menschen, Workshop Pervasive University.

[29] C. Bizer, T. Heath, T. Berners-Lee, “Linked Data: Principles and State of the Art”, 17th International World Wide Web Conference, WWW2008, China, 2008.

[30] R. Fielding, “Architectural Styles and the Design of Network-based Software Architectures”, Dissertation, Irvine, 2000.

[31] E. Wilde, R. J. Glushko, “Document design matters”, Commun. ACM 51, 10, 2008, pp. 43-49, doi: http://doi.acm.org/10.1145/1400181.1400195.

[32] Adobe Flash/Flex, http://www.adobe.com/products/flex/, visited 27.05.2009.

[33] Knol, http://knol.google.com/k, visited 27.05.2009.

[34] C. Kiefer, Quizzer, http://www.d-elina.de/2008/index.php?option=com_content&task=view&id=97&Itemid=64, last update: 2008.

[35] Prototype, http://www.prototypejs.org/, visited 27.05.2009.

[36] Dojo, http://dojotoolkit.org/, visited 27.05.2009.

[37] YUI, http://developer.yahoo.com/yui, visited 27.05.2009.

[38] OpenRDF Sesame, http://www.openrdf.org/, visited 27.05.2009.

[39] Semantic Web Application Framework, http://doi.ieeecomputersociety.org/10.1109/MS.2007.126, visited 27.05.2009.

[40] Swoogle, http://swoogle.umbc.edu/, visited 27.05.2009.

[41] Sindice, http://www.sindice.com, visited 27.05.2009.

[42] Watson, http://watson.kmi.open.ac.uk/WatsonWUI/, visited 27.05.2009.

[43] JPA, EclipseLink, http://www.eclipse.org/proposals/eclipselink/, visited: 13.05.09.

[44] OpenScreen Project, http://www.openscreenproject.org/, visited 27.05.2009.

[45] Apache Tomcat, http://tomcat.apache.org/, visited 27.05.2009.

[46] M. Loukides, „Maven - The Definitive Guide“, O´Reilly Media, Inc., Sebastopol, 2008.

[47] Flickr, http://www.flickr.com/, visited 27.05.2009.

[48] OpenStreetMap, www.openstreetmap.de/, visited 27.05.2009.

[49] Yahoo! Maps, http://maps.yahoo.com/, visited 27.05.2009.

[50] Google Maps, www.maps.google.com, visited 27.05.2009.

[51] OpenID, http://openid.net/, visited 10.04.2009.

[52] FOAF, http://xmlns.com/foaf/spec/#term_name, visited 27.05.2009.

[53] M. Völkel, SparQL Implementations, ESW Wiki, 2008, http://esw.w3.org/topic/SparqlImplementations, visited 27.05.2009.

[54] J. Kumar, “Benchmarking results of mysql, lucene and sphinx”, 2006, http://jayant7k.blogspot.com/2006/06/benchmarking-results-of-mysql-lucene.htm, visited 27.05.2009.