Author's response to reviews

Title: AGUIA: Autonomous Graphical User Interface Assembly for Clinical Trials Semantic Data Services

Authors:

Miria C Correa (mcoelho@mdanderson.org)
Helena F Deus (mhdeus@mdanderson.org)
Ana Tereza R Vasconcelos (atrv@lncc.br)
Yuki Hayashi (hyuki@mdanderson.org)
Jaffer A Ajani (jajani@mdanderson.org)
Srikrishna V Patnana (spatnana@mdanderson.org)
Jonas S Almeida (jalmeida@mdanderson.org)

Version: 3 Date: 30 August 2010

Author's response to reviews: see over
Dear Editor,

Thank you for this opportunity to address the reviewers’ comments and to revise our manuscript. Please find here our point-by-point responses to the reviewers’ comments. We are very grateful to all the reviewers for their comments and for the time they took to experiment with the prototype. We found their suggestions for improvement to be very thoughtful and helpful in conveying how the AGUIA construct explores a representation and modeling framework that is still technologically immature. We note that the profile of RDF/SPARQL-based modeling frameworks has greatly increased over the past few months (see, for example, http://data.gov or http://vivoweb.org). Addressing the comments and concerns of reviewers #2 and #3 was relatively straightforward as they required us to clarify and reinforce the two key features of our research prototype: that it should run entirely on the client’s browser, and that the GUI should cope with fluid schemas. We dealt with the comments from reviewer #1 differently because those comments requested a comparison with other programming/development frameworks, such as Ruby on Rails. We have addressed that critique by ensuring that the research described is that of exploring an automated GUI assembly when the schema is fluid and the computational environment is strictly the "web world" (mostly client-side computation), versus web-based (mostly server-side computation).

Looking forward to hearing from you,

Miria Correa

Reviewer #1 : Prakash Nadkarni

1) The use-case for RDF appears to be that, in situations where database schemas are highly volatile, RDF provides the flexibility of entity-attribute-value (EAV) modeling, where one can simulate virtual schemas in metadata. (RDF is a kind of EAV data store.) In the authors’ scenario, the metadata is used to capture not only the virtual schema but also information about how data is to be presented.

This is correct; however, RDF emphasizes and requires formalisms (URIs) that aim at the global association between data elements. In contrast, EAV is concerned with the local data store. Also, the RDF schema (where S3DB comes in) allows for the definition of generic relationships, which can be used to annotate and generalize individual assertions. As described and referenced in the revised Introduction, this distinction is of critical importance when dealing with fluid schemas.

2) The use of metadata to capture presentation information is not novel: frameworks such as Ruby on Rails (which also support a REST approach) have been doing this for a while: the only difference is that they configuration files rather than RDF stores to persist their metadata. In the reviewer’s own experience, because the metadata schema itself is not volatile, a 3NF relational sub-schema can also be used to capture presentation metadata.
In reply to this comment, we point out that we do not explore a case in which "the metadata schema itself is not volatile." In our examples, the metadata schema does change (initially to our despair), such that 3NF is not flexible enough. Not only does the schema change but different users may want to use their own variants; thus, the schema is extremely volatile. We have edited the Introduction to clarify this point. What drove this research project was actually our dismay at discovering that the metadata schema in these research databases does change as results are processed, or as new people join in, novel relationships are proposed, or as old assumptions are pushed out, etc. An extreme example is the ongoing shift of gene-centric schemas to slicing-variant-centric representations. These changes are usually not reflected directly in the metadata but indirectly in the redesign of clinical trials.

We understand – and share – the reviewer’s admiration for the elegance of Ruby on Rails. However, we feel that the difference pointed out by the reviewer, "that they [use] configuration files rather than RDF stores to persist their metadata," is an important difference! Ruby on Rails is a great place to develop applications for the RDF-based web of linked data. This is widely acknowledged, to the extent that some people have started talking about "RDF on Rails." We did not use this approach because downloading applications or even plug-ins was not an option and we wanted to explore solutions that did not rely on server-side executions of code. With these conditions, one is left with RDF stores and JavaScript coding. We have made some modifications in the Introduction to clarify these constraints to the domain of solutions.

3) > What would make the authors’ paper viable (or not) is not the fact that they use RDF per se, but how they allow a user (who is possibly a non-programmer) to specify various aspects of the desired interface, both at the level of the “class” of data – e.g., chemotherapy, demographics, visits, histology – as well as individual fields within a class – e.g., gender, date of birth, tumor grade. Also, since classes of data are inter-related – e.g., one patient has multiple visits – one typically wants to see summary information of the “many” data – e.g., visits – when looking at a form containing a record of the “one” data – e.g., demographics of a single patient.

The users can and do specify the desired interface, through the use of the eleven components shown in Table 1 of the manuscript. These components can be used according to the needs of each user. As in the example cited by the reviewer, the user can specify a gender field as type “hasOptions,” in which the options are male and female. The data of birth field is of a data type with the action “hasData.” If a patient had several visits, the action “hasMoreFields” assembles a grid to show all visits for this patient at the same time. A similar case can be seen on the Tissue Samples tab of the GI Clinical Trials project, for which a patient can have more than one tissue sample collected and all of the tissue samples can be seen at the same time. This is possible because the collections Sample Information, Frozen Sample, Paraffin Sample and Sample Release are child nodes of the Tissue Samples collection. This was declared in the domain with actions hasMoreFields on the GUI rules collection (instance of GUI rules). We have added that example in the supplementary material; please see Sample Information collection at
The manuscript also includes text on page 14, lines 336-338, which refers the reader to additional supplementary material that explains how to instantiate each graphic component (http://sites.google.com/site/aguiadocumentation/documentation/graphics-components).

4) > This information is simply not present in the paper. Even then, I would have given it a pass if, after inspecting the sample Website (lbl.mdanderson.org/edu), I could see a unifying coherence in the user interface. Unfortunately, after trying out the sample Website, I am convinced more than ever that this interface has not been used for extended periods (if it has been used at all in production) by anyone other than the paper’s authors.

We hope that the extended supplementary material will satisfy the reviewer regarding this critique. In addition, Table 1 shows an example of each graphic component that can be created. The details about how to instantiate each component are contained in the supplementary material

https://sites.google.com/site/aguiadocumentation/documentation/graphics-components.

The reviewer’s inability to see coherence in the interface may be related to the particular way he ran his actions on it. The authors hope that after reading our responses and revisions in the supplementary material and in the manuscript, the benefits brought by this interface will be apparent to the reviewer. This application has been and is being used by users from different departments at MD Anderson Cancer Center, as explained in the text, on page 16, lines 381-385. Some improvements in the application and the addition of new features have been made as a result of the feedback that these users have provided. Nevertheless, there is little we can do about the reviewer’s disbelief that the interface has been used for a sufficiently long period to adequately test it. We do note that some of the authors are clinical researchers and have not written a line of code in their entire lives. On a more even plane, it may be useful to note that the non-programmer users were the ones who specifically asked for each rule. This suggests that maybe they did not have difficulty learning how to use the tools because they were the ones who specified the layout. We note that it was the frequency with which the users re-specified the layout which led us to resort to a more abstract representation of the "GBox" rather than to recode the interface each time. We also note that different users may want different GBoxes and that, short of coding a different interface for each user, we could not have provided such capability without the RDF + js approach. It may also be useful to consider what alternatives the clinical researchers had to achieve this effect, short of having a team of professional programmers on call. The alternative would have been something like Microsoft Excel (they tried!). We all know what sort of mess that causes. Nevertheless, we agree with the reviewer that this is not commercial-grade software, but is a research prototype exploring novel and more abstract formalisms to approach what ultimately is the machine learning challenge of personalized self-assembled interfaces, with the constraints that it has to use the web-browser's JavaScript interpreter and allow the users to continuously modify one or more schemas.
5) The site was clearly test data – patients with the name PatientX, and surgeons with the name Surgeon2. Of course, I don’t expect to see real data at a site (HIPAA considerations), but the data should at least be realistic. While I was able to look at the details of the Histology class (which has the fields Adenocarcinoma subtype, tumor grade, etc.), I could not even look at the details of a single histology record – all I got appeared to be an audit trail of changes made to this class by individual users. Several classes have zero records.

> Needless to say, there seems to be no means of viewing data from more than one class simultaneously - e.g., demographics in a form, and summary histology information in a sub-form/table. Most rival frameworks have this capability.

The fact that the reviewer was unable to see details of a registry reflects not the lack of test data but the particular way he performed the search on the AGUIA. Unlike the structure found in a relational database, in which there is an identifier field for the record, the AGUIA has a main collection that should be linked to other collections to create a bond between them. The main collection takes a linking hub role that is somewhat like a “primary key” for the other related collections. In the study case of gastrointestinal clinical trials, a main collection is Patient ID, as shown in Figure 2. To perform a search that returns all data about a specific patient, the user must include in the search the Participant or MR field, as shown in the figure hereafter. Both fields are items of the Patient ID collection. If one of the two fields is not added to the query, the search will return only patient data that are on the Demographics collection, because this collection contains the Patient Name field.

It is also worth noting that if the user wants to view all collections, it is necessary to select the main collection in the folder tree. This is reason why the reviewer was not able to see more than one “class” (collection) simultaneously.

We found this discussion to be a very useful exercise in anticipating a "relational bias" by some of the readers and have added new text at the end of page 12 and beginning of page 13, lines 302-306, to explain the convergence in user-driven schemas of a main collection concept. The supplementary material also includes a more detailed discussion of this topic (http://sites.google.com/site/aguiadocumentation/documentation/how-to-create-data-ontology/main-collection) and explains how to search using AGUIA (http://sites.google.com/site/aguiadocumentation/documentation/how-to-search).
There is a significant discrepancy between the video and the site. While the video demonstrates the ability to add a record through a forms-based interface, I searched in vain for a means of doing that on the site. (There was a means of uploading tab-delimited files, hardly an effective means of providing interactive data validation, which all of the existing GUI frameworks provide.) Indeed, it is not clear how robust the data-validation component of the framework is. If a video differs from a site, I have to assume that it is the site that represents reality.

We have interpreted the reviewer’s comment as a request for a demonstration of how to insert and update individual records, which was indeed not included in the screencast. The original video demonstrates, with examples, how the eleven graphic components can be created and provides one example of a search. In response to this reviewer, we have added new videos and corresponding written material to the website to demonstrate the functionalities of data insertion (http://sites.google.com/site/aguiadocumentation/documentation/how-to-insert), updating (http://sites.google.com/site/aguiadocumentation/documentation/how-to-update) and search (http://sites.google.com/site/aguiadocumentation/documentation/how-to-search). On page 13, lines 303-306, we added text that refers the reader to the additional supplementary material that explains how to perform a search, or insert or update data using AGUIA.

Please note that all videos are screencasts of using the tool in real time, and that there are no compositions/animations or any uses of mock applications.

In their response to the reviewer’s complaint about the seemingly random order of folders (each folder corresponds to a class of data), the authors stated, “The folder tree now can be sorted by the user through of insertion of a number after the name of collection. If the user doesn’t set any order, the folders will be ordered alphabetically.” I could not do this – presumably some one with administrator privileges could. In any case, re-ordered folders by making a permanent change to the class name is one of the most counter-intuitive (and destructive) ways of reordering data. The logical means of ordering data (which would allow customization by user – the pathologists would want to access different information rapidly than the radiologists – would be to maintain ordering metadata that recorded the preferred order of display. The fact that the authors haven’t considered this leaves me wondering how sophisticated their display rules are. (Individual fields within a class are also required to be prefixed with numbers to force presentation order – thus, the field “tumor grade” is really “03_tumor_grade”. Can’t the authors separate the intended presentation order from the field’s name itself?

We hope that our answers to the previous critiques have shed some light on this concern. The research described here seeks solutions that can cope with multiple fluid schemas in a Web 2.0 environment. The specific case study is an illustration of how AGUIA meets this challenge. In the illustrative scenario defined by the users themselves, they did not want to sort collections and rules by name. They started with a worksheet that should be processed by AGUIA in a way that should preserve their own ordering of each collection and rule in this collection. As in many other real world situations, this order was not a reflection of a logic model but was instead a reflection of the history of this particular community and the way they conceived the relationships between these 31 collections and 189 rules. There was no
request by the users to expand AGUIA to automatically prompt the user for the display order of the 220 fields. It turned out to be more practical to rely on the alphabetic sorting and let the users add a number or a letter before the collection name to control it. The reviewer is nevertheless absolutely right that the future development of AGUIA should expand the possibilities of further tailoring the definition of the GBox. However, the main point we want to demonstrate with AGUIA is that between an RDF-based companion GBox and a generic application written solely in JavaScript, the interaction with knowledge bases can be achieved in a more fluid, personalized, user-driven fashion. We hope that the revised manuscript does a better job of steering the reader away from comparisons with commercial-grade software that do not address these challenges.

8) > Since the display rules were not described anywhere in the paper, I tried inspecting the GUI_Rules folder on this site. I got back an enormous list of rules, but was unable to inspect the details of even one of them – other than the columns ID, “Notes”, Created On, Created By. The essential information – what is the rule’s definition? – was unavailable. How is a power user who is not an RDF guru to go about the task of specifying aspects of the user interface?

Please see the previous response. We are claiming and illustrating that it is possible to derive self-assembled GUIs on the user’s browser that are tailored to a personalized and fluid definition of a GBox. We hope that our intent and the nature of the proposed solution is clearer in this revision of the manuscript. The theoretic and structural description of the graphic modules are more forcefully divided into two collections: GUI rules and GUI actions. The manuscript now expands on what each of the collections contains in a more structured manner. See, for example, at the end of page 10 and beginning of page 11, lines 245-270. Also note that Figure 1 shows the relations with GUI rules and GUI actions collections. The reader is then referred to the expanded supplementary material for more details and a discussion. The URL http://sites.google.com/site/aguiadocumentation/documentation/how-to-create-graphic-module/gui-rules, in particular, shows each rule on the GUI rules collections with an example of how to use each one. The URL https://sites.google.com/site/aguiadocumentation/documentation/how-to-create-graphic-module/gui-action details the GUI actions collection and describes the graphic effect that is elicited by each action.

The user can now also learn how to instantiate each of these rules and manipulate each of the graphic components of the interface. Specifically, see the supplementary material at http://sites.google.com/site/aguiadocumentation/documentation/graphics-components/fields. Please note Figure 5 of manuscript was renumbered Figure 4. It was revised to allow for a general view of the functional architecture of AGUIA, as specifically requested by another reviewer. The representation of the graphic module now contains a list of instances of each one of the rules and actions. We have added text on page 15, lines 359-371, that explains the details of this architecture and text at the end of page 10 and beginning of page 11, lines 248-250, that defines an S3DB collection and rule.
9) > There was a “SPARQL” tab. Is the typical bench scientist or clinician expected to know SPARQL to query the system? I would have expected a query-by-example type interface that generates SPARQL behind the scenes. In summary, this interface fails to meet the most basic usability requirements - the ability to list data in summary and in detail (one record at a time) and allow its interactive editing with validation.

Your point is well taken. We will not try to describe or sell AGUIA as a commercial product. The SPARQL endpoint indeed falls into this category of research features, but with a more mature projection. Unlike our exploratory definition of the GBox, SPARQL is, forgive the pun, completely out of the box. Even initiatives as public as http://data.gov now highlight the SPARQL-based interoperability for distributed knowledge bases. See, for example, the link at the bottom of the demo webapp, http://www.data.gov/semantic/BroadbandAdoptionUrbanVsRural2009. In the research community a similar focus of SPARQL-based interoperability is taking root, in initiatives such as those funded by the NSF and NIH, for example, http://vivoweb.org/ (implementation plan at http://vivoweb.org/files/ImplementationPlan.pdf). Accordingly, the "SPARQL" tab in the S3DB database is there because this database was not created exclusively for AGUIA. On the contrary, S3DB was developed as a knowledge annotation schema not all that different from W3C's SKOS, except in its focus on requirements specific to the management of biomedical content. Please note that a new report was published describing the S3DB core model ("S3DB core: a framework for RDF generation and management in bioinformatics infrastructures", BMC Bioinformatics 2010, 11:387, http://www.biomedcentral.com/1471-2105/11/387). We cite this publication in the revised manuscript.

Although this line of argument is beyond the scope of this manuscript, it is worthwhile to note how AGUIA makes use of the SPARQL endpoint offered by the S3DB. If you dig in the search code, or track the GET calls made by AGUIA (for example using Firefox's firebug, or IE or Chrome native development tools), you will note that the interface indeed generates SPARQL to interoperate with the RDF store. For example, if the user wants to query for the names of female patients who have a specific tumor grade, say G2 moderately differentiated tumors, the only thing the user must do is to add these fields in the query, as shown in the following figure.
If you listen for the HTTP requests, you will see that AGUIA will generate a REST GET call with a SPARQL query:

https://ibl.mdanderson.org/edu/sparql.php?key=xxxxxxxx&query=SELECT * 
?PatientID :R268102 ?Demographics . ?Demographics :R1217829 
?DemographicsPatientName . ?Demographics :R1217849 ?DemographicsGender 
FILTER regex(?DemographicsGender,"^Female") . ?Histology :R1284151 
?HistologyTumorGrade FILTER regex(?HistologyTumorGrade,"^G2 Moderately 
differentiated") . ?PatientID :R1207736 ?PatientIDParticipant .}

We have therefore addressed this particular critique by raising the veil over the use of SPARQL by AGUIA. On page 16, lines 377-380, we have added new text to explain that search, insert and update functions on AGUIA are made through both S3QL and SPARQL queries. We dare to note that the fact that the reviewer did not notice that the unfriendly SPARQL was being used by AGUIA indicates that the SPARQL usability requirement was, in fact, met by this application. Recalling from a previous critique, the use of these features is also described in new screencasts where, accordingly, no emphasis or expertise in SPARQL, nor knowledge of the intricacies of SPARQL is required to perform a search, an edition or an insertion.

10) > I therefore have to judge the paper, not by the authors’ good intentions (including the provision of their effort as open-source), or the fact that they employ RDF or REST
principles, but by the end-results, in terms of a convincing demonstration of what the framework can actually do. Even if they made the case (which they do not) that their framework was mature, they would need to contrast its strengths and limitations with other perfectly good, existing alternatives. (Many of these, such as Ruby on Rails, are also open-source.)

Thank you so much for the time, detail and attention you have given to reviewing and critiquing our manuscript and application. We appreciate your input. The comments have been tremendously useful in our revision of the manuscript. Accordingly, we have now reviewed the text to inhibit readings in which AGUIA is understood as being anything but a working demonstration of the possibility and practicality of personalized self-assembled interfaces that resort to semantic web abstractions and pure JavaScript-coded, browser-based applications. We hope that the revised report describes a domain of solutions that is not in conflict or even in comparison with frameworks such as Ruby on Rails. For example, AGUIA's GBox abstraction could be interpreted by a Ruby application making use of the ActiveRDF library (http://activerdf.org/). That might in fact have been our choice if we were, for example, to develop a heavy duty AGUIA application for mobile devices such as cell phones, in which the browser resources are considerably meager.

Reviewer: Marie-Christine Jaulent
1) > Some improvements have been done to give explanations and document the different resources but however the paper stays difficult to read.

We have now had our manuscript proofread by a technical writer at MD Anderson Cancer Center.

2) > The JavaScript application arrives very late in the paper and don’t see any concrete use case that demonstrate the benefits of the approach.

We agree that this was a major flaw of the original structure and have revised it. After all, this is half the justification for the structure of AGUIA. We are now convinced that without this revision we risk having this manuscript read as a description of a conventional server-side web application. Please note Figure 4 of manuscript was renumbered Figure 5. Figure 5 provides an additional, annotated, concrete snapshot of the practical use of AGUIA in projects conducted in the Gastrointestinal Medical Oncology Department of MD Anderson Cancer Center. The resulting autonomy of the interface assembling application and its ability to deal with volatile – and distinct – GBoxes is now highlighted in the revised manuscript. As a consequence, data can be changed at any moment without any modification of code being necessary. Different users can even have different interfaces generated for the same data simply by invoking distinct GBoxes. Please see our replies to reviewer #1 for a more detailed discussion of how the revised manuscript emphasizes the benefits of AGUIA in dealing with
volatile schemas. See also, for example, the text on page 16, lines 385-392. The abstract also was revised to reinforce emphasis about the benefits of using AGUIA.

3) > An overall picture the functional architecture of the web application will have been useful.

Agreed. We thank the reviewer for this suggestion. Please note Figure 5 of manuscript was renumbered Figure 4. Figure 4 has been revised accordingly to include a general view of the functional architecture of AGUIA. See also new text on page 15, lines 359-371.

Reviewer: Ronald Cornet

1) > The manuscript has improved as it has become clearer what the work was that the authors have done. However, still the structure needs improvement, and there are many spelling errors (which is somewhat disappointing given that this is a resubmission which was proofread according to the authors), which need to be corrected.

Regarding the structure, I suggest a clearer subdivision of the various sections. The part of the introduction starting at line 140 describes how the work was done. The paper would benefit from a more theoretical approach: What do you want? How do you do that? In this case: first, definition of a GBox (reference to 20 (Motik) should be given much earlier). Then implementation of a "rendering engine" (AGUIA). Finally, testing/evaluating the environment with data of 1369 patients.

We thank the reviewer for providing a detailed list of errors. We have corrected them all and in our shame have handed our revised manuscript to a professional writer/editor who is employed at MD Anderson Cancer Center.

As regards the manuscript structure, we have added new text in strategic places, such as at the end of page 6 and beginning of page 7, lines 139-159, to answer the questions "What do you want?" and "How do you do that." Furthermore, we have provided clear definitions of novel concepts, namely GBox, in the manuscript rather than in the supplementary material. The implementation of the rendering engine of AGUIA is included in the Results section and in the subsections "RDF protocol and list of graphic rules" and "JavaScript Application." As suggested by the reviewer, we have added a subsection at the end of the Results that describes the test/evaluation, see pages 18-20, lines 426-483. The abstract itself was revised to reinforce the advantages of using AGUIA.