Flexible DDI storage

Oliver Hopt, Claus-Peter Klas, Alexander Mühlbauer[1]

Abstract

The current usage of DDI is heterogeneous. It varies over different versions of DDI, different grouping, and unequal interpretation of elements. Therefore, provider of services based on DDI implement complex database models for each developed application, resulting in high costs and application specific and non-reusable models.

This paper shows a way to model the binding of DDI to applications in a way that it works independent of most version changes and interpretative differences in a standard like DDI without continuous reimplementation. Based on our DDI-FlatDB approach, shown first at EDDI 2015 & 2016, we present a complete implementation along the use case of a web-based questionnaire editor including sustainable solution for version management and efficient handling of large DDI structures. The user interface is adaptable to different usage scenarios to come. The application supports DDI-Lifecycle from the first question draft and hands over structured (meta-) data to survey institutes and data archives and supports the collaborative questionnaire development for the Pre-election Cross-Section of the German Longitudinal Election Study from early 2017 on.

Keywords

Questionnaire Development, Storage, Compatibility, DDI, Software Architecture

Introduction

DDI is the best known and widely used standard for describing the social sciences data lifecycle from the conceptual start of a “study” to the support of reusing data found in data archives. But the definition of a standard format (together with the implicit data model underneath) is not enough to actually support all the steps in this lifecycle. There is also an enormous need for software systems to gather the metadata to be filled into the standard and to provide it to the scientific and general community.

Making use of DDI in software systems is often a large investment on the implementation side and also on the standardization side. For several software architectural approaches this investment will last until the next version of the standard, until it has to be renewed. This paper presents an storage architecture to work independent of most version changes and interpretative differences in a standard like DDI without continuous reimplementation.

Problem description

The expressive strength of DDI on one hand is a disadvantage on the technical side when it comes to implementing software with and against DDI. The enormous complexity is a matter of cost within classical development, where each class to be supported becomes also a class within the data model. Even those bits from the standard that just bring structure and understandability to the metadata, but have no software-technical necessity have to be implemented.

Concerning the evolution of DDI, we see a constant extension of complexity together with the extension of expressiveness. But the underlying principles of data modeling change on a large scale especially between DDI Codebook and DDI Lifecycle. A good example is the documentation of questions, which are placed as simple <qstnTxt> within variables (<vAr>) in DDI 2.1. In 3.0 they moved into an own package with widely increased details as <QuestionItem> or <MultipleQuestionItem>. And even within the different minor version of DDI Lifecycle, the documentation of questions is not 100% compatible. <MultipleQuestionItem> in 3.0 and 3.1 became <QuestionGrid> and items within those were inline <QuestionItem> and became a set of <Category> within one <GridDimension>.

A more detailed discussion on the technical issues concerning meeting the complexity of and diversity within DDI could be found in [Amin et al. 2011].

To make it even harder for developers, the standard is not always interpreted in the same way by all of its users due to its semantic complexity. This is even the case within institutions using DDI, where not all people have the same understanding of how to use certain fields.

In the following sections of the paper we will first describe our way of storing DDI XML files across versions and across different interpretations which we call dialects. Second, we will describe a portal application to develop and manage questionnaires in, both from the usage perspective and some technical details. Finally, we give a summary and outlook on the next steps for our portal and extended use cases for the DDI-FlatDB.

Making use of DDI-FlatDB

The main idea behind the DDI-FlatDB Architecture is to store the DDI XML files not in a database, but to store them on the hard disk and keep them managed in a version control system, such as GIT. This will eliminate the necessity to develop an entity relationship model for each DDI dialect, DDI version and custom DDI semantics. The file is stored as is. In addition, the version control system provides basic revision and versioning of the DDI files.

In order to enable efficient access to the elements within the DDI files, a customizable split mechanism is provided, which splits each XML file into identifiable XML elements, according to the functional requirements of the application. These elements are then stored in a simple database, currently with one table, which holds the main keys and ids of the element along with its XML content. This database is intended as “proxy only” to provide efficient read and write access and should be configured strictly to functional and application specific requirements.

The XML elements within the database, respective in the file storage, are accessible via a RESTful API, depicted in Figure 1: DDI-FlatDB Implementation.

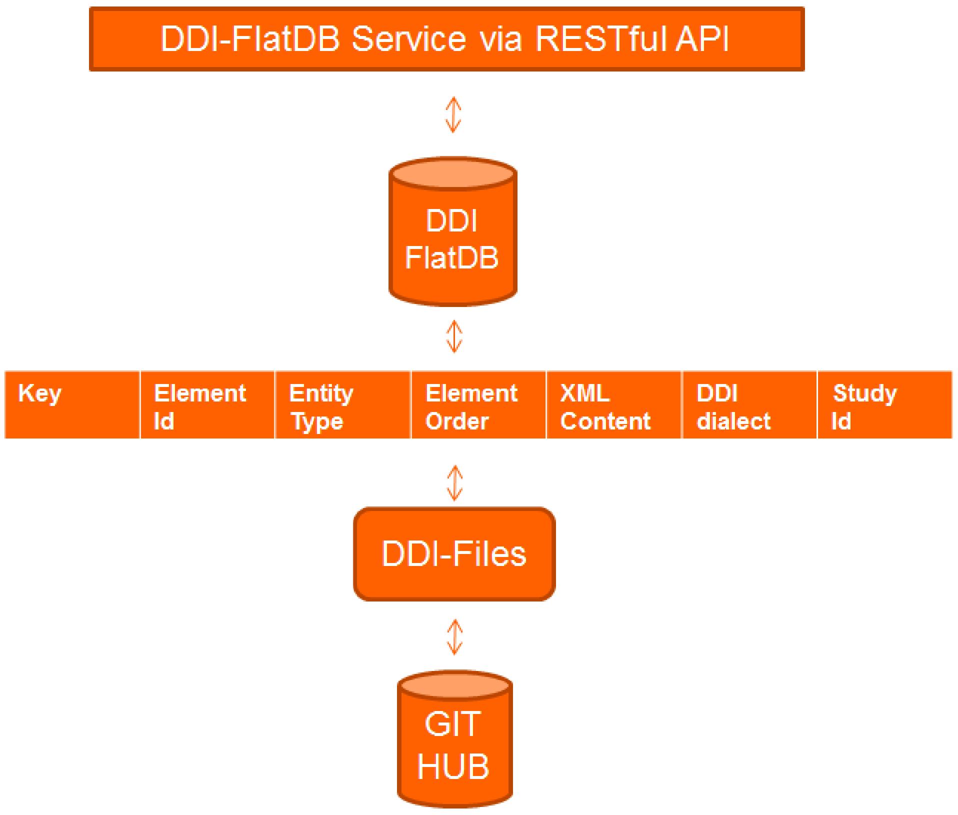


Figure 1: DDI-FlatDB Implementation

The main functionalities provided by the API are

- upload/import and split an DDI XML file with its attached split configuration,
- get element lists (e.g. all studyunits),
- get, set and delete elements.

Upload and split DDI XML document

The upload, split and versioning process is implemented as depicted in Figure 2: Upload, Split and Versioning Process. The initial step is to upload the DDI XML document incl. the application specific split configuration via REST POST. The original document is stored within the flatDB incl. automatic revisioning. The next step is to apply the split configuration, inserting the elements from the split XML document into the flatDB. Now it is possible to use the elements within an application. Through a scheduled poll or event driven service the document is finally stored in the file system and version controlled via GIT.

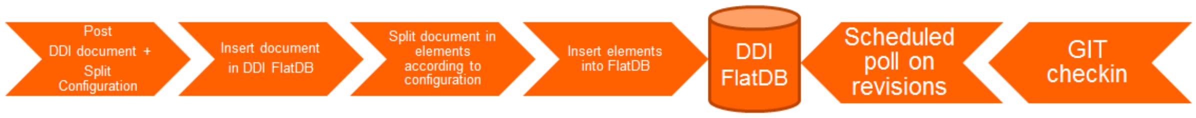

Post DDI document + Split Configuration
Insert document in DDI FlatDB
Split document in elements according to configuration
Insert elements into FlatDB
DDI FlatDB
Scheduled poll on revisions
GIT checkin

Figure 2: Upload, Split and Versioning Process

Example for a simple split configuration

In the following a small example for a split configuration is given to split the "studyunit" into the elements "archive" and "othermaterial", given a DDI3.1 XML document. The main entity in the file is to describe the split paths via XPath expressions. This line also defines the type of the element. The first line in the split configuration gives each element "archive" the type "ddiinstance.studyunit.archive" and is split at "/DDIInstance/StudyUnit/Archive". In addition the identifier path gives XPath location relative to the elements XPath.

The split configuration is used to split and integrate the XML, so the order of the "paths" is important to recreate a well-formed and validated XML file.

```
ddii ddiinstance.studyunit.archive.path = /DDIInstance/StudyUnit/Archive

ddiinstance.studyunit.archive.identifierPath = ./@id



ddiinstance.studyunit.othermaterial.path = /DDIInstance/StudyUnit/OtherMaterial

ddiinstance.studyunit.othermaterial.identifierPath = ./@id



ddiinstance.studyunit.path = /DDIInstance/StudyUnit
```

Figure 3: Example split configuration

Access elements in DDI-FlatDB

The API provides efficient access to lists of DDI element types (e.g. all studyunits) or single elements (e.g. one questionconstruct within a questionnaire) for read, write or delete actions.

In addition all revisions of a document or element can be loaded.

- Fetch lists of elements: <host>/getElementList/<type>
- Fetch single element: <host>/getElement/<elementId>
- Fetch revision of element: <host>/getElementRevisions/<elementId>/<since>

The <elementId> corresponds to the actual identifier within the DDI file. The <type> of the element is derived from the split configuration.

Connecting the DDI-FlatDB to the user interface implementation

As described before, the split configuration defines the elements to be accessible from the DDI-FlatDB. In a user interface the mapping from XML elements to Java objects is needed. As we wanted to be flexible and again follow the functional dependencies, we decided to implement the necessary objects as simple Java Beans. To be independent of the DDI dialect, we provide for each element a configuration, how and where to read and write the values. The following code snippet gives a simple example, how this is done for the property label from the entity QuestionItem:

```
Que QuestionItem.Label.ldep=true

QuestionItem.Label.multi=false

QuestionItem.Label.complex=false



gesisquestionnaire32.QuestionItem.Label.path=/d:QuestionItem
```

Figure 4: Property file for entity parsing

The path to the label value is given by an XPath expression. As the label can be given by multiple languages, the phrase %lang% is replaced by the current content language set in the application context. In addition the key "ldep" (language dependent) is set for processing. Thus DDI-FlatDB can handle different language versions. Complex entities, such as multiple publications with own attributes can also be instantiated, but need a more complex configuration. Within the Java Bean, the configuration of "QuestionItem.Label" fills the values of java object attribute label, using Java reflection and the setter and getter pattern.

In the next section we describe our use case project, the questionnaire and study editor, depending heavily on the DDI-FlatDB approach to also be used in the future, when new DDI versions come up. In the best case, we only need to adapt the above described configuration files, instead of remodeling the database.

Questionnaire and study editor

Following one of the main goals, implementations around the DDI-FlatDB are driven by their use cases, the first step towards the questionnaire and study editor was to collect requirements from the actual users of this tool. Initially it was projected to support the German longitudinal election study (GLES) with management facilities for their questionnaires for the German federal election in 2017. The resulting workflow, depicted in Figure 5: Questionnaire workflow looks very similar to the general process of creating questionnaires [Blumenberg et al. 2015].

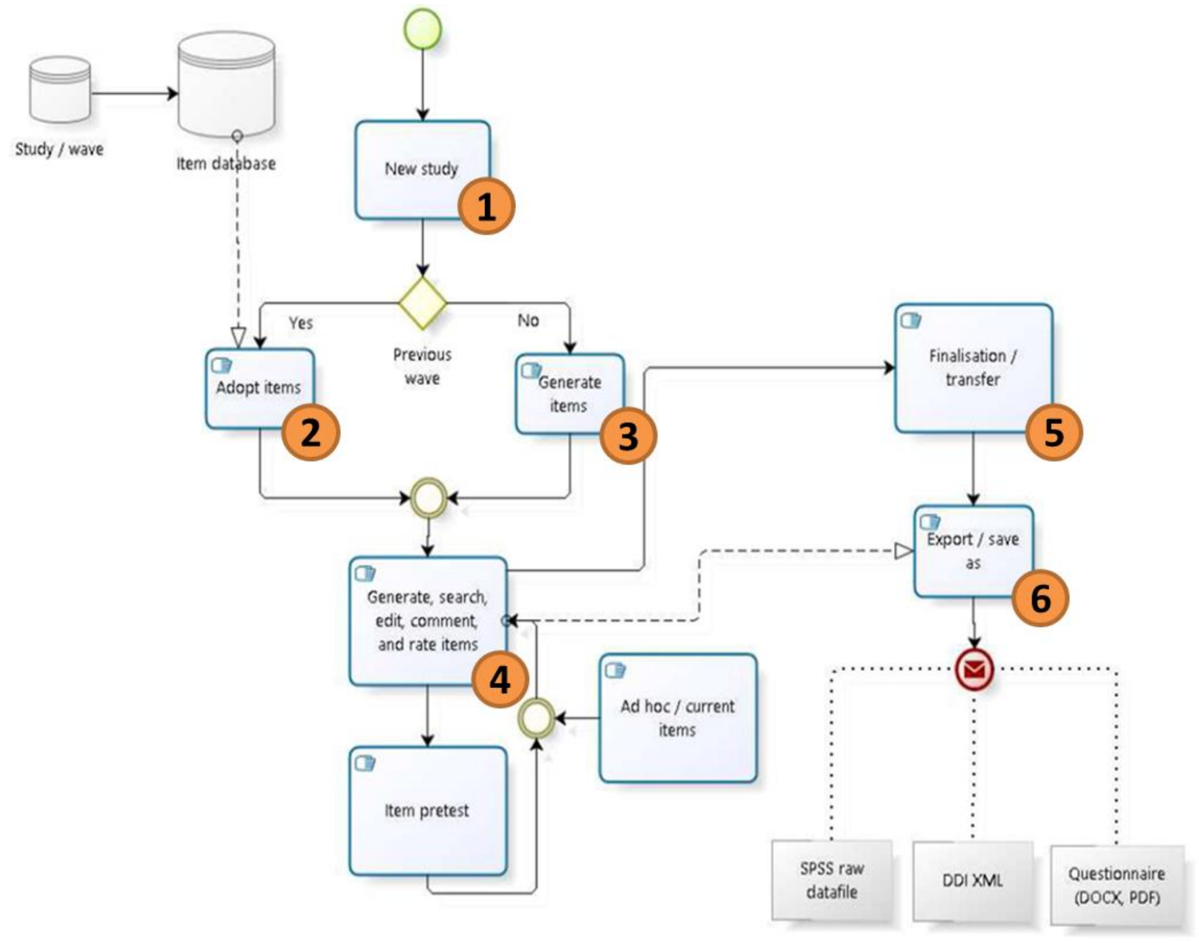


Figure 5: Questionnaire workflow

Caused by the need for a new study (1) the researchers want to either start with a complete or partial copy of a questionnaire from a previous wave (2) or with a blank questionnaire to be filled with new questions (3). Now, the actual collaborative and iterative process starts to fine tune the instrument (4), including editing, discussing and to rate existing questions as well as the creation of new questions both from scratch and as import from the collection of questions already existing inside the question database.

After finalizing the definition of the new questionnaire (5), it should no longer be edited by the researchers in general. Only a study administrator is then able to do error corrections on the existing content. This study administrator will also perform a delivery to a survey institute as a set of possible exports (6). These exports include a customizable questionnaire document and an unfilled file of any statistical software including the defined variables. If survey institutes will be able to process DDI in the future, the application will be able to hand over such files instantly as it is directly relying on DDI files. Along with the definition of the questionnaires, we included a study editor to document the study itself.

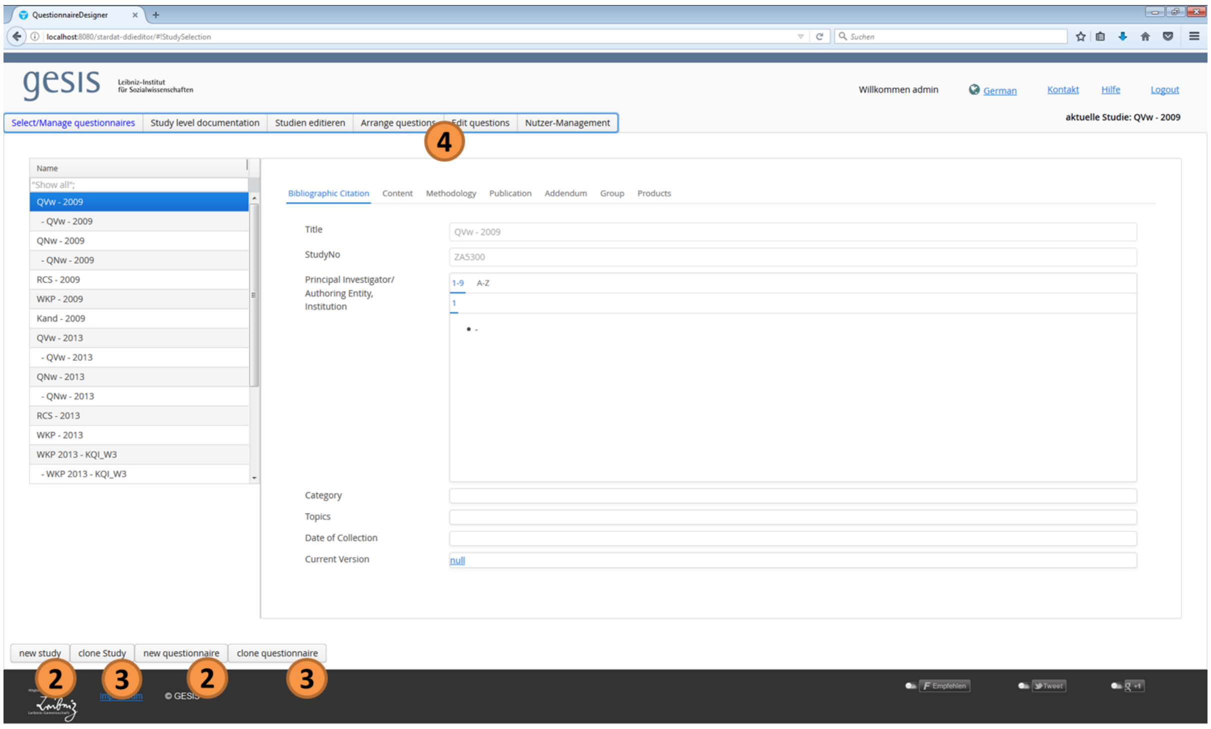
The numbering of these steps (1-6) will reappear later in this paper to associate these steps with the actual implemented functionality.

The current mapping into DDI is based on version 3.2. This includes both a split configuration on the level of the DDI-FlatDB and a mapping onto the used Java entities. These entities and the resulting user interfaces are modeled along the data model behind DDI lifecycle. The consequence of this approach is that some splits may not work in the same manner in other versions of DDI. The used structure for items within question grids for example cannot be mapped even into DDI 3.1 without using the same XML snippet to fill several different entities. To avoid dirty reads, legacy data could be viewed within the described portal but not edited. For reuse in new questionnaires the entity classes could be used to convert copies to DDI 3.2.

The portal currently contains 13 individual studies with 30 questionnaires containing about 4000 questions only for the German election study alone.

Portal components

The current editor application contains only functionality that needs authorization. Therefore, every visit of the portal will lead first to a login form. After authentication, the user is directed to a questionnaire/study selection view (see Figure 6), containing all studies with respect to his/her group affiliation and according to the access rights, a set of management functions. These consist of creating a new study (3) or clone an existing study (2) and create a new questionnaire (3) or clone an existing questionnaire (2). The distinction between studies and questionnaire is necessary due to the requirement within GLES to have a series of questionnaires included in one data set. After choosing a questionnaire or study, the top menu can be used to either edit the study level documentation or the questionnaire content (4).



aktuelle Studie: QVw - 2009

Name
"Show all";
QVw - 2009
- QVw - 2009
QNw - 2009
- QNw - 2009
RCS - 2009
WKP - 2009
Kand - 2009
QVw - 2013
- QVw - 2013
QNw - 2013
- QNw - 2013
RCS - 2013
WKP - 2013
WKP 2013 - KQI_W3
- WKP 2013 - KQI_W3

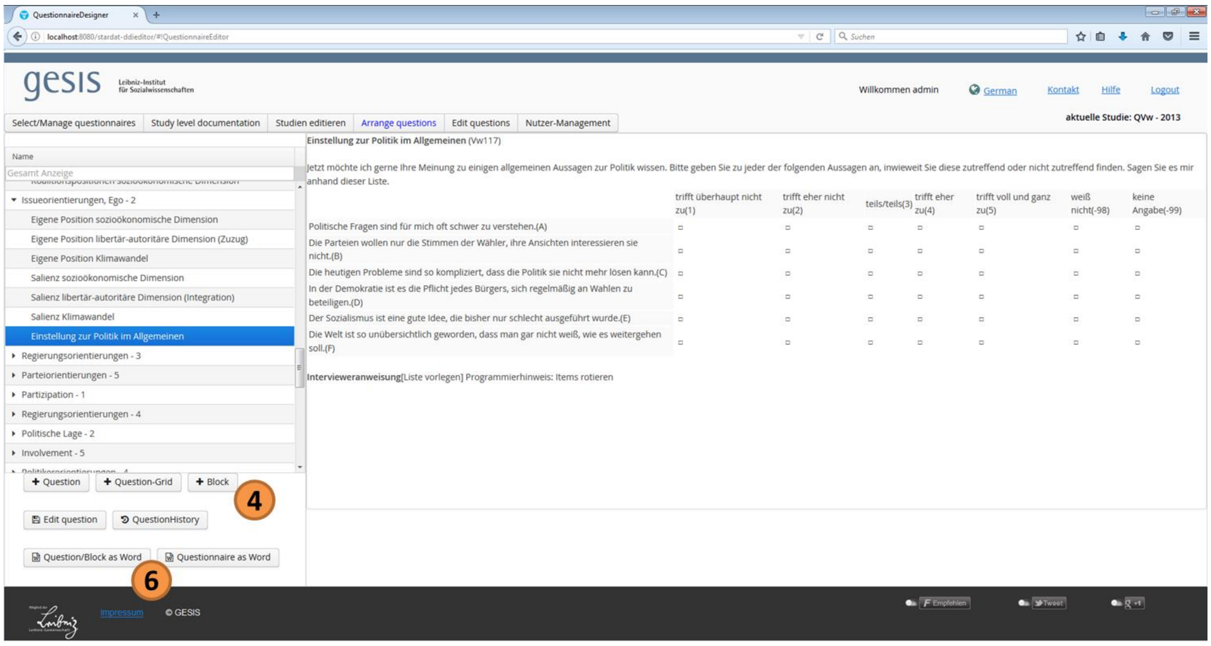
Title	QVw - 2009
StudyNo	ZA5300
Principal Investigator/ Authoring Entity, Institution	1-9 A-Z 1 • -
Category	
Topics	
Date of Collection	
Current Version	null

Figure 6:Study and Questionnaire Selection View

The portal offers two slightly different views for editing questions. The first one offers a list of all questions contained in a questionnaire sortable by different criteria. One of these consists of personal tags to be associated with questions to mark them as of individual interest or with importance (rating). It is also possible to filter within each of these criteria.

The second view, show in Figure 7, offers a hierarchical list of logical blocks (DDI:sequences) and questions in the straight forward flow of the questionnaire. Within this view, it is not possible to sort, because the order of the overview table is used as the arranging device of the questionnaire. Researchers can drag and drop or move single questions or entire blocks of questions through the questionnaire flow, so that the overall questionnaire can be rearranged (4). Both views contain functionalities for creating and editing questions (4) and for exporting certain questionnaire elements or even the entire questionnaire (6). The tree-table form for logical blocks (sequences) is covering not much more than label and description to be stored within <Sequence>. The <ControlConstructReference> elements included are managed by the tree view on the left side of the main screen.

The center of this main view always contains a preview of the selected element. In case of a question grid (or item battery), the preview will give an impression, how the element would look like in the final questionnaire. Simple question look similar but the preview of a sequence just consists of its definition part and a list of all contained elements.


QuestionnaireDesigner
localhost:8080/stardat-ddieditor/#!QuestionnaireEditor
Suchen
gesis Leibniz-Institut für Sozialwissenschaften
Willkommen admin
German
Kontakt
Hilfe
Logout
Select/Manage questionnaires
Study level documentation
Studien editieren
Arrange questions
Edit questions
Nutzer-Management
aktuelle Studie: QVw - 2013
Name
Gesamt Anzeige
Issueorientierungen, Ego - 2
Eigene Position sozioökonomische Dimension
Eigene Position libertär-autoritäre Dimension (Zuzug)
Eigene Position Klimawandel
Salienz sozioökonomische Dimension
Salienz libertär-autoritäre Dimension (Integration)
Salienz Klimawandel
Einstellung zur Politik im Allgemeinen
Regierungsorientierungen - 3
Parteiorientierungen - 5
Partizipation - 1
Regierungsorientierungen - 4
Politische Lage - 2
Involvement - 5
Question
Question-Grid
Block
4
Edit question
QuestionHistory
Question/Block as Word
Questionnaire as Word
6
Impressum
© GESIS
Empfehlen
Tweet

Einstellung zur Politik im Allgemeinen (Vw117)

Jetzt möchte ich gerne Ihre Meinung zu einigen allgemeinen Aussagen zur Politik wissen. Bitte geben Sie zu jeder der folgenden Aussagen an, inwieweit Sie diese zutreffend oder nicht zutreffend finden. Sagen Sie es mir anhand dieser Liste.

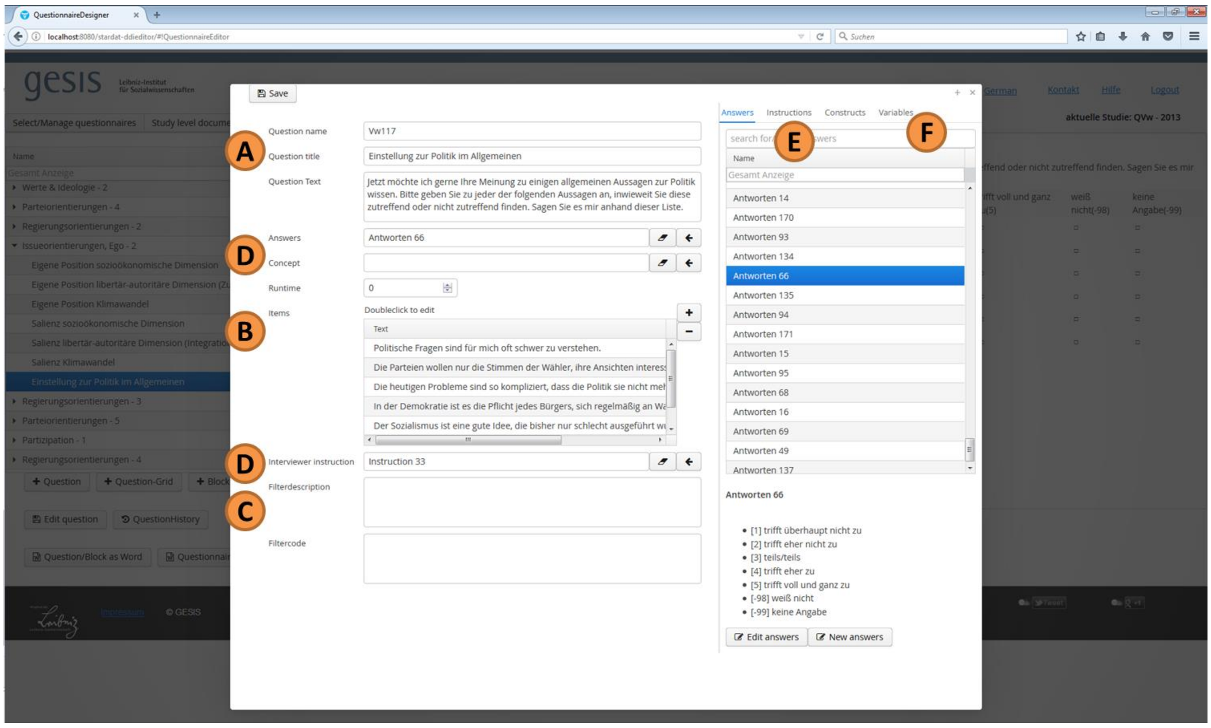
	trifft überhaupt nicht zu(1)	trifft eher nicht zu(2)	teils/teils(3)	trifft eher zu(4)	trifft voll und ganz zu(5)	weiß nicht(-98)	keine Angabe(-99)
Politische Fragen sind für mich oft schwer zu verstehen.(A)	☐	☐	☐	☐	☐	☐	☐
Die Parteien wollen nur die Stimmen der Wähler, ihre Ansichten interessieren sie nicht.(B)	☐	☐	☐	☐	☐	☐	☐
Die heutigen Probleme sind so kompliziert, dass die Politik sie nicht mehr lösen kann.(C)	☐	☐	☐	☐	☐	☐	☐
In der Demokratie ist es die Pflicht jedes Bürgers, sich regelmäßig an Wahlen zu beteiligen.(D)	☐	☐	☐	☐	☐	☐	☐
Der Sozialismus ist eine gute Idee, die bisher nur schlecht ausgeführt wurde.(E)	☐	☐	☐	☐	☐	☐	☐
Die Welt ist so unübersichtlich geworden, dass man gar nicht weiß, wie es weitergehen soll.(F)	☐	☐	☐	☐	☐	☐	☐

Intervieweranweisung[Liste vorlegen] Programmierhinweis: Items rotieren

Figure 7: Question View

The editing of questions is described along the form for question grids (or item batteries) as it is more or less an extension of the field set available for simple question items (see Figure 8). Concerning the mapping into DDI 3, this form consists of several parts.

- The first part contains the essential question elements like name, title and question text (A).This information is stored within <QuestionGrid> (or <QuestionItem>).
- The items of the grid (B) are defined individually for every question grid. As a result of the user requirements, no reuse of item sets is wanted. This results in a pair of <CodeList> and <CategoryScheme> referenced from one <GridDimension>.
- Incoming filter definitions (C) are stored within *Universe*. As this is referenced from the <QuestionConstruct> that maps the grid (or question) into the questionnaire flow, reuse of these definitions is possible. The user requirements contained the request to not manage this information independently but to integrate it into individual questions.
- Answer categories, concepts and interviewer instructions (D) are also referenced from <QuestionGrid> (<QuestionItem>) and <QuestionConstruct>. They are accessed on right side of the form (E) both for selecting the used instances and for managing/editing.
    - Answer categories are stored as pairs of <CodeList> and <CategoryScheme> each.
    - Interviewer instructions go straight forward into <InterviewInstruction>.
    - Concepts are managed as <Concept> within a <ConceptScheme> that is placed outside of the questionnaire file, so that the concepts are consistent throughout the entire GLES study series.
- Variables (F) can also be edited, but as this portal targets to a state before an actual data set is produced/survey is conducted, the variables are only defined by name and label, e.g. for hand over to the survey institution in an statistical file.



Question name: Vw117

Question title: Einstellung zur Politik im Allgemeinen

Question Text: Jetzt möchte ich gerne Ihre Meinung zu einigen allgemeinen Aussagen zur Politik wissen. Bitte geben Sie zu jeder der folgenden Aussagen an, inwieweit Sie diese zutreffend oder nicht zutreffend finden. Sagen Sie es mir anhand dieser Liste.

Answers: Antworten 66

Concept:

Runtime: 0

Items (Doubleclick to edit)

Text
Politische Fragen sind für mich oft schwer zu verstehen.
Die Parteien wollen nur die Stimmen der Wähler, ihre Ansichten interess
Die heutigen Probleme sind so kompliziert, dass die Politik sie nicht meh
In der Demokratie ist es die Pflicht jedes Bürgers, sich regelmäßig an Wa
Der Sozialismus ist eine gute Idee, die bisher nur schlecht ausgeführt wu

Interviewer instruction: Instruction 33

Filterdescription:

Filtercode:

Name
Gesamt Anzeige
Antworten 14
Antworten 170
Antworten 93
Antworten 134
Antworten 66
Antworten 135
Antworten 94
Antworten 171
Antworten 15
Antworten 95
Antworten 68
Antworten 16
Antworten 69
Antworten 49
Antworten 137

Antworten 66

- [1] trifft überhaupt nicht zu
- [2] trifft eher nicht zu
- [3] teils/teils
- [4] trifft eher zu
- [5] trifft voll und ganz zu
- [-98] weiß nicht
- [-99] keine Angabe

Figure 8: Question Edit View

"External" components

Following the idea of micro services, functionality that covers information surrounding the DDI structure is stored in separate services. Our portal currently makes use of services for:

1. User authentication - To support all tools within the same tool chain with a consistent user management, this is externalized.
2. Tagging of DDI elements - For several reasons, private and public tagging can support the productivity within large (meta-) data collections. We developed a service that associates tags with identifiers so that the tags could always be linked to the resources behind the identifiers.
3. Export to Word - This service will evolve to a general export mechanism based on word templates to be filled with the content from DDI elements. Currently it supports questionnaires with a fixed template.

Summary and outlook

The application of the DDI-FlatDB architecture described in this paper is not only implemented with the questionnaire editor, but also is used within our automatic codebook report generation service and applied to the study editor (see Figure 9 blue boxes). In addition we will adopt further editor tools, like the variable documentation. And besides the editor tools, we plan to provide search and browse portals on the documented studies.

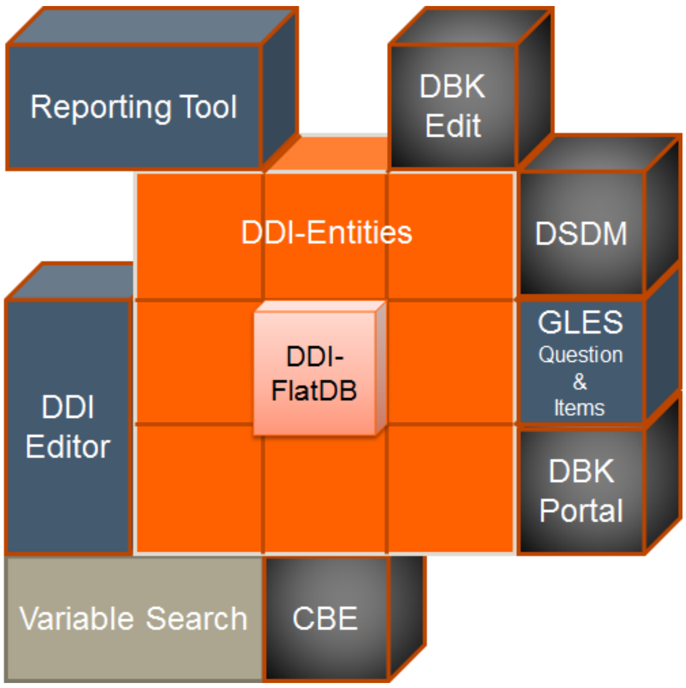


Figure 9: Overview of Services based on DDI-FlatDB

We have achieved up to this point a complete application from storage to a user interface fully based on DDI (within one year development time), with the following features:

- collaborative and web-based,
- not fixed to a specific version or dialects of DDI,
- includes revision and version handling,
- includes user management and access rights.

For the future versions of the DDI-FlatDB an extension is planned to explicitly instantiate the links within DDI, e.g. external links between studies and study waves as well as internal links between questions and variables to be published as Linked Open Data. This extension will provide faster access to entities and provide search and browse functionality. For the questionnaire editor, we will go live in April and will extend the functionality further to meet the user requirements, like more collaborative functions such as a discussion platform. The DDI-FlatDB will be published as open source.

References

Amin, Alerk, Barkow, Ingo, Kramer, Stefan, Schiller, David, Williams, Jeremy. 2011. “Representing and Utilizing DDI in Relational Databases”. DDI Working Paper Series (Other Topics) DDI Alliance. http://dx.doi.org/10.3886/DDIOtherTopics02

Blumenberg, Manuela S., Wolfgang Zenk-Möltgen and Claus-Peter Klas. 2015. "Implementing DDI-Lifecycle for Data Collection within the German GLES project." EDDI15 – 7th Annual European DDI User Conference, Kopenhagen.

Klas, Claus-Peter, Oliver Hopt, Wolfgang Zenk-Möltgen and Alexander Mühlbauer. 2015. "DDI-Flat-DB – a lightweight framework for heterogeneous DDI sources" EDDI16 – 8th Annual European DDI User Conference, Cologne.

Klas, Claus-Peter, Oliver Hopt, Wolfgang Zenk-Möltgen and Alexander Mühlbauer. 2015. "DDI - FlatDB: Efficient Access to DDI." EDDI15 – 7th Annual European DDI User Conference, Kopenhagen.

[1] Oliver Hopt, Claus-Peter Klas, Alexander Mühlbauer are all affiliated with GESIS - Leibniz-Institute for the Social Sciences in Germany. Oliver Hopt are contact author at oliver.hopt@gesis.org.