IADIS International Journal on WWW/Internet
Vol. 17, No. 2, pp. 56-70
ISSN: 1645-7641

SEMANTICAL ENRICHMENT OF WEB USER INTERFACES IN THE CROWD

Claudia Steinberger and Joachim Frießer
Department of Applied Informatics, Universität Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria

ABSTRACT

Structured data on the Web have become very important in recent years and offer machine-readable semantics of the data contained. Schema.org is the most recognized vocabulary for structuring data on websites and serves beyond that also to organize and query Google's knowledge graph. However, Schema.org is mainly used today to semantically describe real-world entities and their relationships to one another. To describe the functionality and handling of websites has not been the key application area of Schema.org yet, though it also includes an appropriate vocabulary. In addition, most Schema.org tools are limited in terms of the Schema.org classes and properties they offer and do not support the semantical enrichment of web user interfaces.
In this article we motivate the potential of the sematic enrichment of web user interfaces regarding their functionality and handling and present possible application areas to consume these structured data. We analyze the requirements on structured user interface data and investigate the suitability of Schema.org as a vocabulary to fulfil them. Since there are hundreds of classes and properties in Schema.org, we present a conceptual model of Schema.org classes and properties that are appropriate to enrich web user interfaces. We investigate to what extent interactive elements on HTML websites or applications can be automatically mapped to our conceptual model elements. As a result, we present a method to semi-automatically produce structured user interface data and illustrate our approach with a continuous use case. As a proof of concept, we introduce *Schemator*, a comfortable structured user interface data tool supporting our method, which can be used to crowdsource knowledge about the functionality and handling of websites and applications.

KEYWORDS

Semantic Web, Web User Interface, Schema.org, Structured Data, Crowdsourcing, Knowledge Base

1. INTRODUCTION

Over the past decade, amazing progress has been made in enriching websites with structured data that provide a machine-readable semantics of the data they contain (Guha et al. 2016) (Lehmann et al. 2015) (Ringler et al. 2017). In general, structured data simply stands for the

systematic structuring of data in order to be found better. In the Web, structured data appear in texts, like the source code of a website or in knowledge bases consisting of structured records. The two approaches may also be combined as a text may be enriched with semantic markup that identifies mentions of entities from a knowledge base. Moreover, several knowledge bases with different schemata may be combined (Bast et al. 2016).

Therefore, structured data does not form a single knowledge base on the Web, since there is no uniform schema to tag the data. Given the heterogeneity of contents on the Web, it seems illusionary to establish a standard for everything and to expect everyone to use this standard. Thus, to structure data on the Web, many vocabularies have been developed (Vandenbussche 2017). One approach to improve the homogeneity of structured data in the Web is to encourage contributors to reuse existing vocabularies as much as possible. A recent approach in this area is Schema.org (Mika 2015).

Schema.org has proven to be the most widely acknowledged open vocabulary for structured data on the Web, in email messages and beyond (Mika 2015) (Guha et al. 2016). The big search engine providers have been the main promoters of Schema.org with the original aim to improve the display of search results, making it easier for people to find the right web pages and to feed their knowledge bases (Paulheim 2017).

Schema.org provides a vocabulary that is compact and easy to use to describe “things” on the Web and to make them interoperable and understandable for automatic processing. Better positions and representations in organic search results, knowledge cards or widespread online map services in turn motivate website designers and operators to provide structured data on their websites. Today over 10 million websites use Schema.org to markup some of their content (https://schema.org). In practice, structured data are mostly used to describe content, namely items of the universe and their properties. Functionality and handling of websites and applications have not been the application focus of Schema.org yet, though it also includes an appropriate vocabulary. In addition, most Schema.org plugins or tools are limited in terms of offered Schema.org classes and properties they offer and are not appropriate to describe web user interfaces.

This article deals with the following research questions: (1) what interests exist to have access on structured user interface data, (2) what are requirements on structured user interface data exist and what Schema.org classes and properties are appropriate, (3) who shall contribute knowledge on user interface functionality and handling and (4) how can structured user interface data be created and made available with the help of a suitable tool.

To find answers to this research questions, this article discusses reasons, why structured user interface data can be helpful and analyzes requirements for structured data. We explain our research process to investigate the suitability of Schema.org as a vocabulary to fulfil our requirements. We start with the investigation of elements and relations of web user interfaces and their semantics, which we intent to make machine-readable. As there exist hundreds of classes and properties in Schema.org, we present a conceptual model of appropriate Schema.org classes and properties to annotate this semantics. We investigate to what extent interactive elements on HTML websites or applications can be automatically mapped to our conceptual model elements.

As a result, we present our method to semi-automatically produce, validate, store and publish semantic annotations of web user interfaces using the Schema.org vocabulary. As a proof of concept, we present *Schemator*, a tool that allows users in the crowd to contribute, collect and maintain structured data on websites and applications. To illustrate our approach, we use a continuous use case.

Possible application areas of structured user interface data are the conversational handling of websites applications using intelligent voice assistants or chatbots, workflow management, the search on the Web or for special functionalities. Another use case shows how the intelligent assistance systems HBMS (Michael et al. 2018) consumes structured user interface data to customize its personalized context model (Michael, Steinberger 2017).

The article is structured as follows: Chapter 2 gives an overview of structured data in the Web, introduces Schema.org and mentions actual producers and consumers of structured data on the Web. Chapter 3 investigates the suitability of Schema.org to semantically enrich user interfaces in the Web and introduces our research method to produce structured user interface data. Chapter 4 introduces the *Schemator*, a prototype that we have developed as a proof of concept to support the production process of structured user interface data in the crowd. Chapter 5 summarizes our findings and gives an outlook to identified further research challenges.

2. STRUCTURED DATA ON THE WEB - PRODUCERS AND CONSUMERS

In recent years, impressive advances have been made in enriching the Web with semantics to provide a machine-readable "meaning" of data. In doing so, structured data have been used to extract structured content from websites. Structured data can be embedded in the source code of websites (Bizer et al. 2013), where they remain hidden from the human reader and are only processed by crawlers or other “intelligent agents” to classify, interpret and process them. Thus, the main consumer of structured data on the Web are machines, not humans. They automatically use embedded structured data e.g. to build up their search engine index or to mine semantic data for their knowledge database, which acts afterwards as a central storage. Some of these knowledge bases are open for all to use and can be queried using an API or SPARQL endpoints (Yu, 2014). Their main goal is to store millions of entities and reliable, associated facts about those entities. Some of the more prominent players in development include Google Knowledge Graph and Knowledge Vault, Bing's Satori, DBPedia or Wikidata (Paulheim 2017) (Färber et al. 2018)(Lehmann et al. 2015).

To structure data on the Web, many vocabularies have been developed (Vandenbussche 2017). But semantic interoperability and the exchange of data with an unambiguous meaning play an important role. One approach to improve the homogeneity of structured data in the Web is to encourage contributors to reuse existing vocabularies as much as possible. Today, Schema.org is the most acknowledged vocabulary to structure data on websites. Schema.org is a collaborative community project promoted by the big search engine providers with the mission to create and maintain a shared vocabulary for structured data used on websites, in email messages, and beyond (Guha et al. 2016) (Hepp 2015). Schema.org structured data can be embedded in the source code of websites encoded in RDFa, Microdata or JSON-LD (Sporny et al. 2014) (Ronallo 2012) (Yu 2011). From there it can be interpreted and individually used by any tool or service (Mika 2015). Schema.org also finds application in the field of knowledge bases and enables e.g. the inclusion of information in the Google Knowledge Graph. The API of the Google Knowledge Graph uses standard Schema.org types and is compliant with the JSON-LD specification (GNG 2019). Google uses its Knowledge Graph e.g. to generate knowledge cards shown in the search console (Paulheim 2017).

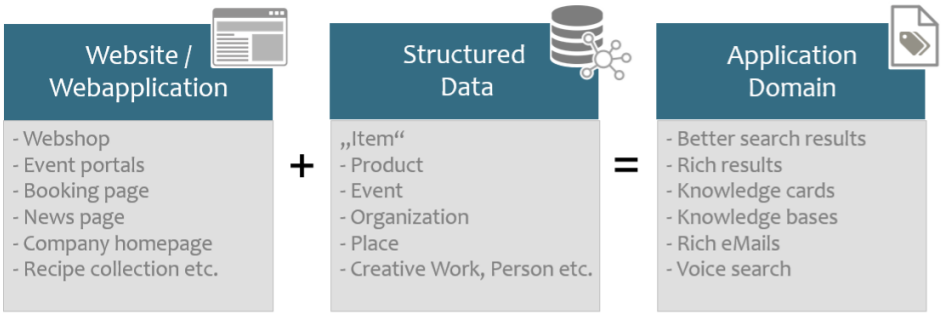

Figure 1. Schema.org structured data on the Web

Today, Schema.org structured data are mainly consumed by search engines to automatically browse websites with high speed and accuracy and to take over search efforts for humans (see Figure 1, application domain). The annotation with structured data increases the chance of a website to appear as a rich result, for example a rich snippet, a featured snippet or a quick answer.

As a result, Schema.org structured data are mainly produced and provided on websites in the context of semantic search engine optimization for content types with mostly direct or indirect commercial relevance like e.g. products, places, events, videos, persons, organizations and more (see Figure 1, structured data). Although Schema.org vocabulary is commonly used for the enrichment of “items”, websites and applications are more than just content – they offer many ways to take actions. Schema.org offers a vocabulary to describe web user interface elements and actions in a structured way (Mika 2015), but structured data describing user interfaces can hardly be found today neither in the source code on websites or applications nor in knowledge bases (Simsek et al. 2018).

Web content enrichment is complicated and costly, but website designers and operators have been motivated in return for better search engine ranking and results. To produce structured data, web designers must deal with the following challenges: the vocabulary to use, to create the desired annotations and to represent, store and publish these enrichments. The demand for tools to produce structured data has grown. There exist a lot of annotation tools and plugins with different levels of automation (Webpals 2019) (WPLeaders 2019). Most of them are not widely used because they are embedded in a CMS that enforces access rights to the CMS backend and usually only support certain Schema.org classes. Some tools are strictly decoupled from the underlying websites and generate, and store annotations separately from the content. However, the complexity of the offered vocabulary often overburdens the average producer of structured data. To counteract this (Khalili, Auer 2013) (Kärle et al. 2017) have worked on solutions for the creation, publication and distribution of semantic annotations of content in an easier and more intuitive way. In contrast to our approach, they did not focus user interfaces in their work.

Moreover, other intelligent agents started to consume Schema.org annotations. The Schema.org markup "Speakable" for instance enables voice assistants to reproduce special extracts of the content of a website linguistically on demand. Annotations of content and actions in rich emails can help them to stand out from the rest in the inbox and to enable the user to call go-to actions very easily (e.g. to order an item, to check in a flight, to reset a password) or to summarize and to highlight key information in the inbox app.

To the best of our knowledge, there is no approach semantically enriching the functionality and handling of user interfaces of websites or applications. The machine-readable 'meaning' of relevant interactive elements like sign in, share, order, browse orders, comment, go to homepage, change language or edit the profile would enable new possibilities for new intelligent agents. Possible application areas are voice assistants for a conversational handling of websites and applications, web application workflow management agents, online help agents, intelligent assistance systems that are interoperable with websites and applications (Steinberger, Michael 2018) or the possibility to search web sites and applications for certain functions (see Figure 2).

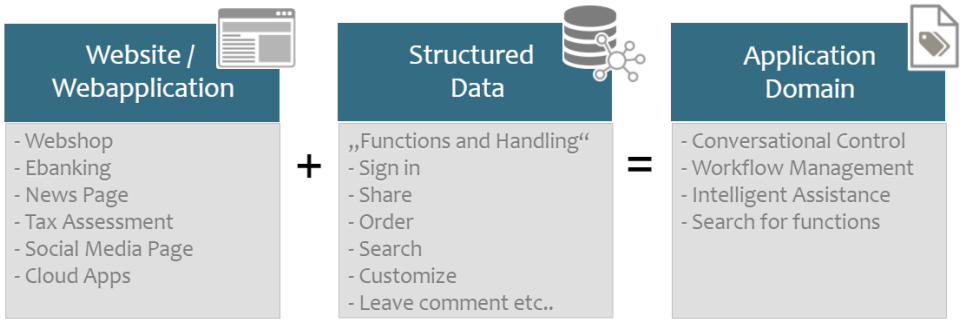

Figure 2. Semantically enriched web user interfaces

With our approach, we want to semi-automatically produce, validate, store and publish semantic enrichment of web user interfaces using the Schema.org vocabulary.

3. SEMANTIC ENRICHMENT OF WEB USER INTERFACES

Figure 3 summarizes our approach to enrich user interfaces with structured data and to store and manage these data in an open knowledge base. On the right-side, Figure 3 includes the research method we have applied and, on the left, it presents the production method for structured user data we propose. We do our investigations on three levels: level I treats the website or application user interface, level II the source code of the website and level III the structured user interface data production. This chapter deals with our research method whereas chapter 4 focuses on the production method.

At level I of our research method, we start the requirements for structured user interface data and try to find ways to meet them. First, we examine the elements and relations of a web user interface and its semantics, which we intent to make machine-readable. We select typical website or application use cases like the homepage of our University, a well-known online shop or an online banking application. Based on these use cases, we identify and categorized those characteristics and possible user interactions that we wanted to enrich with structured data so that we are able to consume them later in conversations with intelligent personal assistants, chatbots or intelligent assistance systems (Michael et al. 2018) (Steinberger, Michael 2018).

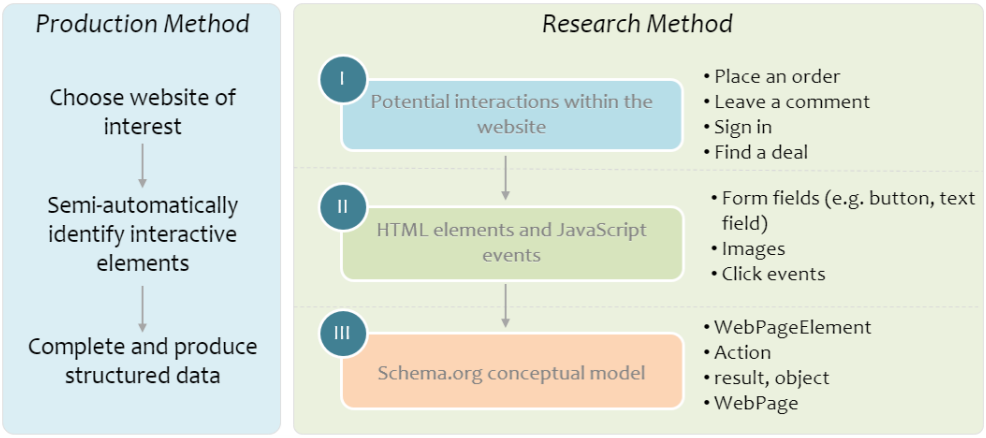

Figure 3. Production and Research Methods to produce semantic user interface data

To illustrate our approach, we use a continuous use case. Figure 4 shows a small fragment of the Amazon.com landing page, which is one of our use cases on level I to investigate elements and relations of web user interfaces and their semantics. We are going to take this use case throughout this article. It is essential on level I to focus on the basic characteristics of the landing page and its potential interaction elements like to find deals, to change the language, to filter categories, to register at the website or to search a product. Based on these interaction elements we characterize the types of the triggered actions, like *find*, *change*, *register*, *order* and to describe the objects and results, an interaction produces (e.g. call of a subpage). We also want to be able to describe the handling of potential interaction elements using media like text, image or video.

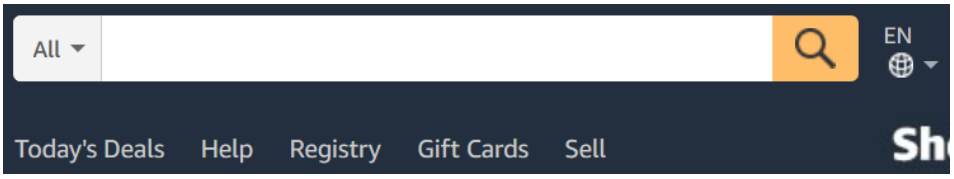

Figure 4. Amazon.com Use Case

The next challenge in our research is to identify classes and properties to semantically describe the characteristics and interaction elements found on level I with the vocabulary of Schema.org. Our investigations are represented as level III of our research method in Figure 3. Schema.org covers a large catalog of classes and properties, but only a small excerpt of the vocabulary turned out to be necessary to describe the identified elements.

Based on the requirements collected on level I we develop a conceptual model of the necessary excerpt of Schema.org classes (see Figure 5). We describe a website or application using the Schema.org class *WebApplication*. *WebApplication* offers the properties *name*, *description*, *keywords*, *image* to describe the required characteristics. A *WebApplication* can include several webpages, what can be described using the property *hasPart*. *Each* webpage can be described using the Schema.org class *WebPage* with its properties *name*, *description* and *headline*.

Schema.org knows several different subclasses of *WebPage*, like *SearchResultsPage*, *ProfilePage*, *QAPage*, *ItemPage*, *ContactPage* and more, what is also helpful in our context (Krutil et al. 2012). *A WebPage* can include multiple instances of the class *WebPageElement*, representing user interaction elements on the weppage. *WebPageElements*, as *WebPage* and *WebApplication*, are all subtypes of *CreativeWork* and therefore inherit the properties *name* and *description*. Schema.org does not support many different subclasses of WebPageElement. Thus, it is not possible to distinguish between elements like buttons, icons, sliders etc. on class level. This differentiation is only possible within the scope of *description* property. To describe the handling of a web user interface, we link *MediaObjects* and the more specific types *AudioObject*, *VideoObject* and *ImageObject* to a *WebPageElement* by applying the property *associatedMedia*.

Every associated *WebPageElement* has a *potentialAction* property linking it with an *Action*. Each *Action* is characterized by a *name* and a *description*. Schema.org knows more than 100 different subclasses of the class *Action* (see http://schema.org/docs/full.html). A subset of these subclasses like e.g. *SearchAction, FindAction, RegisterAction, ApplyAction* or *SearchAction* works well to tag our identified user interactions. Every Action can be associated via an *object* property with an entity of type *Thing* and via a *result* property with a *WebPage* again.

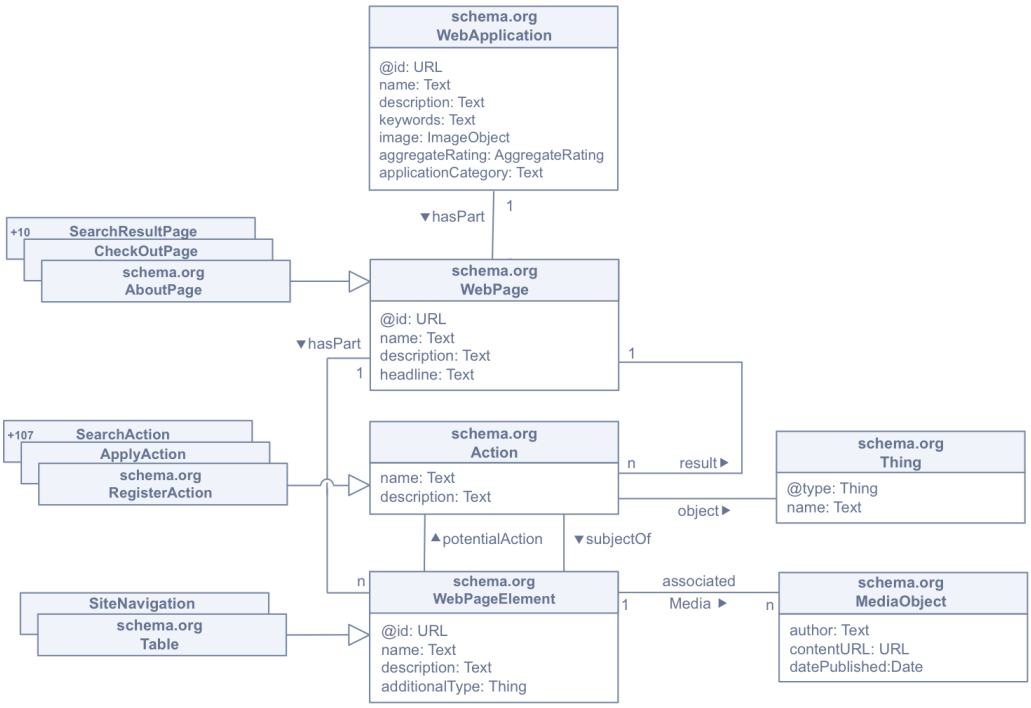

Figure 5. Conceptual model of Schema.org excerpt necessary for user interface enrichment (Frießer 2019)

With this conceptual model in mind our next step is to produce structured user interface data and store it in a knowledge base. We expect his knowledge base to use standard Schema.org types and to be compliant with the JSON-LD specification.

Figure 6 shows simplified structured user interface data of the Amazon.com use case represented in JSON-LD. To produce such structured data as easily as possible, it is desirable to map HTML user interaction elements as automatically as possible to classes and properties of our conceptual model. For that purpose, we investigate on level II of our research method what information from the source code of these web pages can be used automatically to create structured user interface data (see Figure 3, level II).

Thus, we analyze the HTML elements and Javascript events on our use case websites or applications to find out, what semantic information about an identified interaction element can be retrieved from the document object model (DOM). Not every clickable HTML element on a website or application is relevant for us. Fortunately, some of the big companies like Apple, Google, Mozilla and Microsoft worked on the development of the HTML living standard (WHATWG) and introduced a concept named "inertness": If an element is identified as *inert*, the user agent (e.g. the browser) hast to ignore this element for user interaction (e.g. text search or selection). As a result of their work, we can identify a number of interactive HTML elements focusing only those covering at least an activation behavior as specified in (WHATWG): "*change*", "*click*", "*contextmenu*", "*dblclick*", "*mouseup*", "*pointerup*", "*reset*", "*submit*" and "*touchend*" (Frießer 2019).

Our investigations show that the required structured data cannot be reliably and completely extracted from the identified HTML elements on websites or applications. Some naming conventions have been specified in the semantics section of (WHATWG), which state that the name of a HTML tag can have a certain meaning and a clue for the purpose, e.g. the "*addr*" tag is used for an address. Such conventions cannot be reliably adhered to. Even the label or rather a text can be a hint for the functionality of e.g. button, but if an icon is used, a machine won't be able to guess or "see" what the purpose of the control is.

In summary, knowledge gained from the DOM is enough for an initial draft of a website's structured user interface data but has to be checked and supplemented manually. Figure 6 shows the JSON-LD representation of an excerpt of the structured user interface data enriching the use case shown in Figure 4 with the focus on the interaction element represented by a globe icon to change the language of Amazons landing page.

```
{"@context": "http://schema.org",
  "@id": "https://www.amazon.com",
  "type": "WebApplication",
  "name": "Amazon",
  "keywords": "online shopping, prime, kindle, alexa, deals of the day",
  "description": "Shop Amazon's best offers on products that ship to over 100 countries
   around the world through Amazon Global.",
  "image":
  { "@type": "ImageObject",
    "contentUrl": "https://images-eu.ssl-images-amazon.com/images/...468502409_.png"
  },
  "aggregateRating":
  { "@type"     : "AggregateRating",
    "ratingValue" : "3",
    "reviewCount" : "10"
  },
  "hasPart":
  [{
      "@type"   : "WebPage",
      "name" : "Landing page",
      "description": "Browse for offers and buy items online.",
      "headline"        : "Amazon.com: Online Shopping for Electronics, Apparel,...",
      "hasPart":
      [{"@type" : "SiteNavigationElement",
              "name" : "Button to change language",
              "description" : "Click on this button in order to switch the used language.",
              "associatedMedia" :
              {"@type" : "ImageObject",
                "contentUrl" : "https://m.media-amazon.com/images/...6fe68._V2_.png",
                "author" : "Amazon",
                "datePublished" : "2019"
              }
              "potentialAction" :
              { "@type" : "ApplyAction",
                "name" : "switch language",
                "description" : "switch language of webpage",
                "object": {"type": "Language",
                           "name": "German"}
              }
          }
      ]
    }
  ],
  "applicationCategory" : "Webshop",
  "operatingSystem" : "Multiplatform"
}
```

Figure 6. JSON-LD structured user interface data of the launch site of Amazon.com (excerpt)

Chapter 4 now takes a closer look onto the tool support for the semi-automatic production of structured web user interface data and defines the user roles and the architecture of the *Schemator* platform.

4. ENRICHING WEB USER INTERFACES IN THE CROWD

Based on the results in chapter 3 as a proof of concept we now introduce *Schemator*, our tool to support the semantic enrichment process of web user interfaces.

4.1 Schemator

Schemator is a comfortable semi-automatic web application that allows users to enrich the user interface functionality and handling of specific websites or web applications. *Schemator* serves also as an open knowledge base to store and manage the produced structured web interface data. Figure 7 shows the roles that users can assume working with *Schemator*: (1) *Tagger*, (2) *Content Admin*, (3) *Data Consumer* and (4) *Model Designer*.

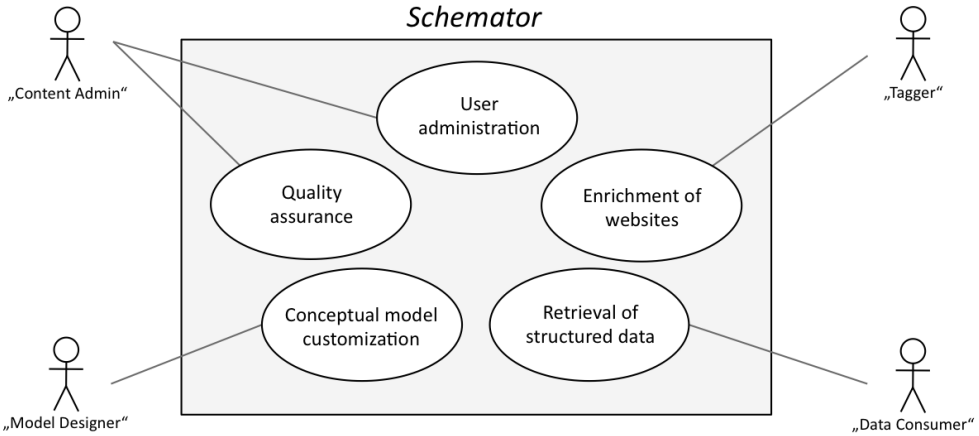

Figure 7. Use case diagram of *Schemator*

The (1) *Tagger's* main task is to annotate user interfaces of websites or applications and to produce or structured user interface data in a semi-automatic way. With his knowledge about the functionality and handling of a website or application and the way a user typically performs certain tasks there the *Tagger* performs the production method presented in Figure 3:

After a website or web application of interest has been selected or registered, the Tagger can view and access its webpages in "browse mode" embedded into *Schemator* in a graphical representation. The website or application under consideration is loaded visually into the *Schemator* and the underlying HTML code is scanned (see Figure 8). To edit existing or add new semantics, the *Tagger* can switch from "browser mode" to "annotate mode". Interactive elements and their properties are listed now in the *elements catalog* on the left side of the *Schemator* user interface. The detected landing page properties and the properties of their interactive elements are displayed now and can be selected and enriched with structured data according to our conceptual model in Figure 5. By clicking on an interactive element directly or on an item in the elements catalog, *Schemator* highlights the control (e.g. an input field or a button) and scrolls it into the *Tagger's* view if necessary. Now the *Tagger* can focus the interactive element and edit the corresponding structured data on the right side of the *Schemator* interface. The same is done for subpages.

An advantage of using *Schemator* is that *Taggers* do not need to have any previous knowledge of the Schema.org classes and properties they have to use. The input fields displayed on the right are dynamically generated based on the conceptual model presented in chapter 3, Figure 5.

The vision of *Schemator* is to enrich web user interfaces in the crowd. Already available structured user interface data contributed by others can be read, extended and adapted. Comments can be left, the quality of the structured interface data can be rated and erroneous annotations can be fixed and reported. Crowdsourcing knowledge on a large scale and in a reasonable period requires large crowds. This makes it impossible to rely only on experts, review each volunteer or manually review individual contributions (Heindorf et.al 2016). Working with the crowd therefore means trusting them. Projects like Wikipedia and WikiData prove that you can trust the crowd to build and maintain reliable knowledge bases in the freedom that anyone can edit anything.

Schemator follows this assumption in the current version too but to prevent vandals from abusing the *Schemator*, the (2) *Content Admin* has the possibility to validate annotations, to grant or deny user rights or to ban unwanted websites.

The (3) *Data Consumer* retrieves structured user interface data from the *Schemator* knowledge base via an API or SPARQL endpoint in form of JSON-LD documents. In fact, the structured data can be also embedded as an external link target of a script tag.

Finally, the role of the (4) *Model Designer* is to customize the conceptual model of the Schema.org vocabulary that *Schemator* is supposed to support. In the current version, the conceptual model presented in Figure 5 is supported but Schema.org also evolves further and *Schemator* is flexibly customizable this way. If any adaption is necessary, e.g. new and useful action types or new properties are of interest, the Model Designer can customize this modification without changing the code.

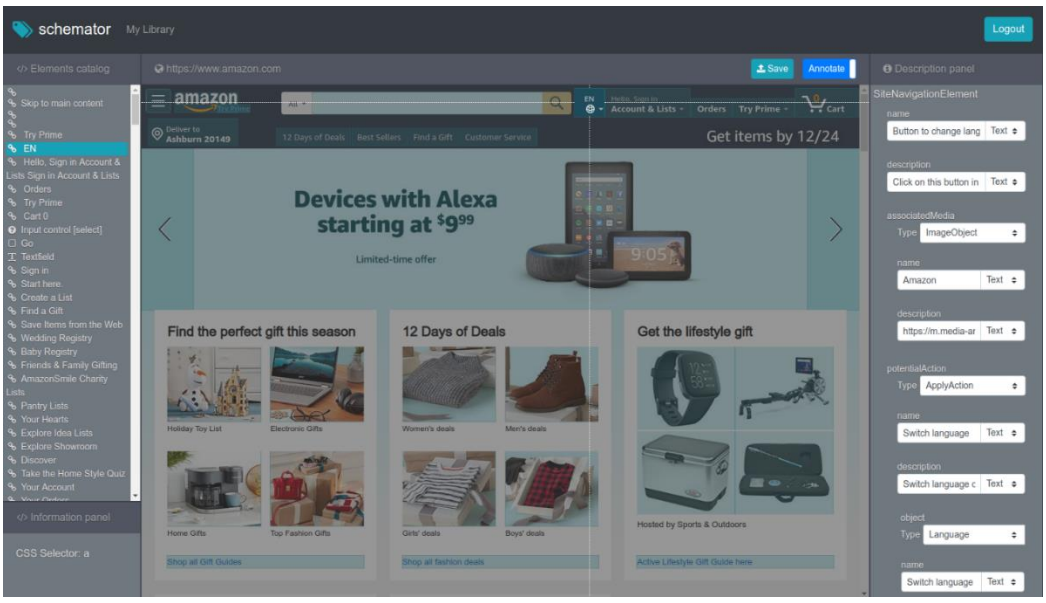

Figure 8. User interface of *Schemator* showing Amazon.com in the "annotate" mode

As a proof of concept, a first prototype of *Schemator* was implemented in a first version as a web application in the context of a master thesis at the Universität Klagenfurt and covers the main functionality to enrich website or applications (Frießer 2019). Figure 8 shows a snapshot, where *Schemator* is used by a logged in *Tagger* in annotate-mode to enrich the user interface of

the Amazon.com website. The scanned user interface elements of Amazon.com are listed on the left, the structured user interface data are displayed on the right side of Figure 8. To focus and annotate a special element, the Tagger can use the reticle.

4.2 Use Case of a Structured User Interface Data Consumer

A data consumer, like an intelligent agent, workflow system or conversational agent can retrieve and use the structured user interface data from the open *Schemator* knowledge base. Figure 9 shows a scenario, where the active assistance system called Human Behavior Monitoring System (HBMS) consumes *Schemator's* structured user interface data. HBMS is a system, which monitors elderlies in their physical environment using different sensors. HBMS can gain context knowledge in this way (Michael, Steinberger 2017) and supports the elderly in their daily activities (e.g. order food, create tax return). Giving support means to help people to remember how they once performed an activity by reactivating already existing memory anchors, what makes it easier to remember situations and handlings. Thus, HBMS supports the autonomy of a person with decreasing memory.

As digitalization of daily life progresses, it becomes increasingly important to help elderlies to use their needed websites or applications. Web user interfaces cannot be treated as elements in the physical user environment. Instead of installed location based, body or object based sensors, web user interfaces possess potential interaction elements. The environmental context knowledge about functionality and handling of websites or applications needed to support a specific user can be imported flexibly from the *Schemator knowledge base* into the personalized context model of the HBMS (figure 8, upper left corner) (Steinberger, Michael 2018).

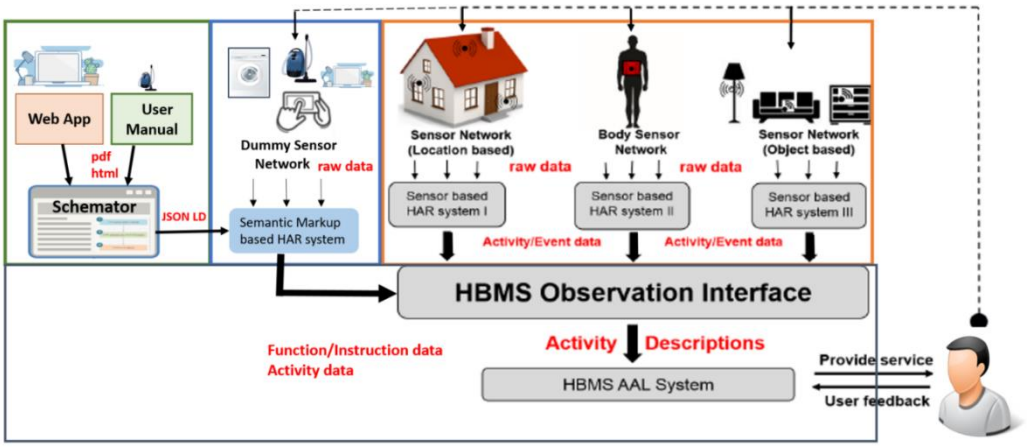

Figure 9. Semantic markup interoperability scenario with HBMS
(adapted from (Steinberger, Michael 2019))

4.3 Schemator Architecture

Schemator relies on client/server architecture. As Figure 10 shows, due to security reasons the web client does not directly interact with a website or application under consideration. The server component additionally acts like an "in between web proxy". The *Content Checker* scans

the website or application and only if it considered as safe, then *Schemator* acts like a real webserver proxy and saves the downloaded files in the *Website Cache Storage*. *Schemator's* main data representation and interchange format is JSON-LD, even for the representation of the Schema.org conceptual model as presented in chapter 3. Because of the strong use of JSON-LD, MongoDB (Chickerur et al 2015), a document-driven database is used for persistent data management on the server component. The task of the client component is to recognize the potential "tag-worthy" interactive elements of the website.

Some common frameworks like *Bootstrap* and *jQuery* are used for *Schemator's* user interface and business logic (https://getbootstrap.com/; https://jquery.com/), the latter framework is mainly responsible for the interactive control detection. Finally, the *LocalStorage* as defined in (WHATWG) is used to temporarily save valuable information of already annotated items. This architecture allows a disaster recovery of the data if the browser crashes during the tagging process. Moreover, functionalities to retrieve structured user interface data and to customize the conceptual model are implemented but have not yet been integrated into the *Schemator* web application. A next *Schemator* version is currently under development and focusing structured data quality and system security.

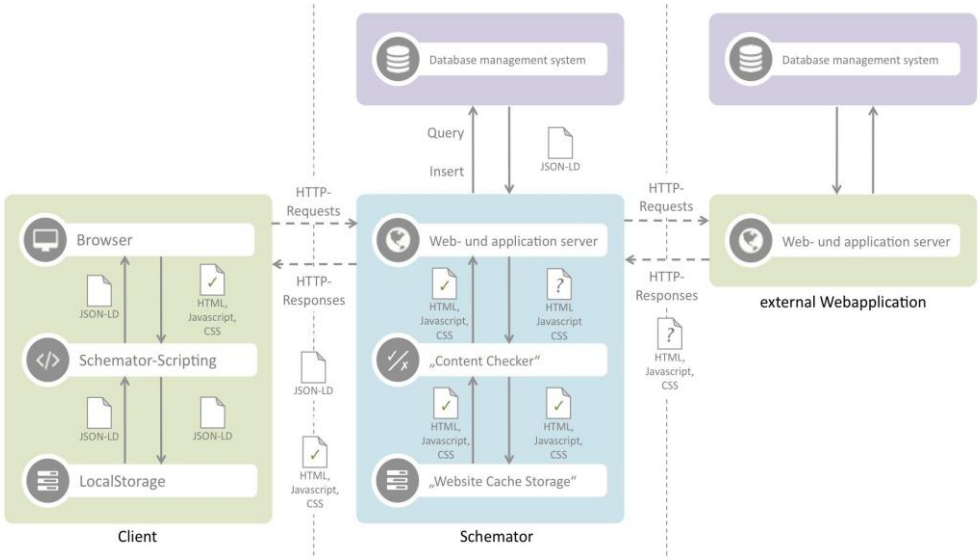

Figure 10. System architecture of the *Schemator* following (Frießer 19)

5. CONCLUSION AND OUTLOOK

In this article, we motivate the importance of structured user interface data and showed that the Schema.org vocabulary is suited also to semantically enrich the user interface of websites or applications. We present a conceptual model of appropriate Schema.org concepts to annotate interaction elements and show possibilities and limits to map interactive HTML elements to these Schema.org model elements.

As a proof of concept, we present *Schemator*, a prototype to support the semantic enrichment process of web user interfaces in an easy and intuitive way. The existing version of the *Schemator* has not yet completely fulfilled all use cases presented in Figure 7, but it covers the core functionality. The enrichment process works, and structured data represented in JSON-LD can be retrieved and consumed by intelligent agents. *Schemator* is not only a platform for generation, editing and storage of annotations but also a means to validate and publish them. By this way, *Schemator* offers a free and open knowledge base of structured user interface data collected in the crowd, separating structured user interface data from the websites they describe. Despite the success of the free-to-edit model, all knowledge bases that rely on it are plagued by vandalism. Vandalism has been around ever since knowledge crowdsourcing emerged. Given the importance of knowledge bases to modern information systems, there exists a significantly higher demand on the integration of structured knowledge bases (Heindorf et al. 2016). In future, we want to continue developing the next *Schemator* version, collect structured user interface data of selected websites and applications and evaluate the *Schemator*. In addition, we want to investigate how to detect vandalism in *Schemator*.

We also plan to use *Schemator* beyond the enrichment of web user interface to semantically describe the functionality and handling of appliances. His flexible conceptual model enables it to broaden the application area of *Schemator*. In this context, we plan to produce and administrate structured user manual data (Steinberger, Michael 2019).

REFERENCES

Bast, H., Buchhold, B. and Haussmann, E., 2016. Semantic search on text and knowledge bases. *Foundations and Trends in Information Retrieval*, *10*(2-3), 119-271.

Bizer, C., Eckert, K., Meusel, R., Mühleisen, H., Schuhmacher, M. and Völker, J.,2013. Deployment of rdfa, microdata, and microformats on the web–a quantitative analysis. In *International Semantic Web Conference* (pp. 17-32). Springer, Berlin, Heidelberg.

Bizer, C., Heath, T., & Berners-Lee, T. (2011). Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts* (pp. 205-227). IGI Global.

Chickerur, S., Goudar, A. and Kinnerkar, A., 2015. Comparison of relational database with document-oriented database (mongodb) for big data applications. In *2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA)* (pp. 41-47). IEEE.

Färber, M., Bartscherer, F., Menne, C. and Rettinger, A., 2018. *Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago*. Semantic Web, 9(1), 77-129.

Frießer, J., 2019. *Semantische Aufbereitung von Benutzeroberflächen von Webapplikationen für die aktive Assistenz mit HMBS*, diploma thesis, August 2019.

GNG, Google Knowledge Graph Search API, https://developers.google.com/knowledge-graph, (last access 2019/12/01).

Guha, R. V., Brickley, D. and Macbeth, S., 2016. *Schema. org: evolution of structured data on the web*. Communications of the ACM, 59(2), 44-51.

Heindorf, S., Potthast, M., Stein, B. and Engels, G., 2016. Vandalism detection in wikidata. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (pp. 327-336). ACM.

Hepp, M., 2015. *The web of data for e-commerce: Schema. org and GoodRelations for researchers and* practitioners. In International Conference on Web Engineering (pp. 723-727). Springer, Cham.

Kärle, E., Şimşek, U. and Fensel, D., 2017. *semantify. it, a Platform for Creation, Publication and Distribution of Semantic Annotations.* arXiv preprint arXiv:1706.10067.

Khalili, A. and Auer, S., 2013. *Wysiwym authoring of structured content based on schema.org*. In International Conference on Web Information Systems Engineering (pp. 425-438). Springer, Berlin, Heidelberg.

Krutil, J., Kudělka, M. and Snášel, V., 2012. Web page classification based on Schema. org collection. In *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)* (pp. 356-360). IEEE.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., ... and Bizer, C., 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, *6*(2), 167-195.

Michael, J. and Steinberger, C., 2017. Context Modeling for Active Assistance. In *ER Forum/Demos* (Vol. 1979, pp. 207-220).

Michael, J., Steinberger, C., Shekhovtsov, V. A., Al Machot, F., Ranasinghe, S. and Morak, G., 2018. *The HBMS story*. Enterp. Model. Inf. Syst. Arch, *13*, 345-370.

Mika, P., 2015. *On Schema.org and why it matters for the web*. IEEE Internet Computing, 19(4), 52-55.

Paulheim, H., 2017. *Knowledge graph refinement: A survey of approaches and evaluation methods*. Semantic web, 8(3), 489-508.

Ringler, D., & Paulheim, H. (2017, September). One knowledge graph to rule them all? Analyzing the differences between DBpedia, YAGO, Wikidata & co. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)* (pp. 366-372). Springer, Cham.

Ronallo, J., 2012. HTML5 Microdata and Schema. org. *Code4Lib Journal*, (16).

Şimşek, U., Kärle, E. and Fensel, D., 2018. *Machine readable web apis with schema.org action annotations.* Procedia Computer Science, 137, 255-261.

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M. and Lindström, N., 2014. JSON-LD 1.0. *W3C Recommendation*, *16*, 41.

Steinberger, C. and Michael, J., 2018. *Towards cognitive assisted living 3.0*. In 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) (pp. 687-692). IEEE.

Steinberger, C. and Michael J., 2019. *Using Semantic Markup to Boost Context Awareness for Assistive Systems* in Feng Chen, Rebaca I. García-Betances, María Fernanda Cabrera-Umpiérrez, Liming Chen, Chris Nugent: Smart Assisted Living, Springer, 2019, 227-246.

Vandenbussche, P. Y., Atemezing, G. A., Poveda-Villalón, M. and Vatant, B., 2017. *Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web*. Semantic Web, 8(3), 437-452.

Webpals. https://www.webpals.com/seo/15-free-schema-markup-tools/ (last access 2019/12/05).

WHATWG. https://whatwg.org/ (last access 2019/05/24).

WPLeaders.https://wpleaders.com/wordpress-schema-plugins/ (last access 2019/12/05).

Yu, L., 2014. *A developer's guide to the semantic Web*. Springer Science & Business Media.