Can Clinicians Create High-Quality Databases? A Study on A Flexible Electronic Health Record (fEHR) System

Ritu Khare
ritu@ischool.drexel.edu

Yuan An
yan@ischool.drexel.edu

Il-Yeol Song
isong@ischool.drexel.edu

Xiaohua Hu
thu@ischool.drexel.edu

The iSchool at Drexel
Drexel University
Philadelphia, PA-19104, USA

ABSTRACT

Clinicians are becoming increasingly dependent on health information technologies (HIT) in their daily activities, like data collection. However, currently, most HITs are vendor designed systems, which are often inconsistent and inflexible with respect to the needs of the clinicians. Consequently, time and again, the HITs are found to be unfit for the healthcare workflow. A better HIT design is to empower the clinicians with the ability to modify system functionality as per their needs. In this paper, we propose a flexible Electronic Health Record (fEHR) system, which allows clinicians to build new templates/forms for data collection over an existing EHR system through a user interface. The system automatically translates the forms to underlying databases while shielding the user from knowing the technical details. A key contribution is that the generated databases are high-quality with desirable properties. To test the system usability, we conducted a user study with clinicians working in a nurse-managed health services center. The participants performed the given tasks with 100% effectiveness, within a short span of time, in all but one case; and exhibited an improvement in their understanding of the system. Our study demonstrates that the fEHR system has the potentials of incorporating flexibility into HITs to make them more effective and efficient for healthcare. We show that the fEHR is a favorable environment for clinicians to develop and improve their need-modeling skills.

Categories and Subject Descriptors

H.5.2 [**Information Systems**]: Information Interfaces and Presentations—*User Interfaces*; H.2.1 [**Information Systems**]: Database Management Systems—*Logical Design*

IHI'10, November 11–12, 2010, Arlington, Virginia, USA.

General Terms

Algorithms, Design, Human Factors

Keywords

Clinician, Database, EHR, Flexibility, Forms

1. MOTIVATION

Clinicians are becoming increasingly dependent on health information technologies (HIT) in their daily activities. Primarily, a clinician is involved in collecting customized data related to patients, diseases, treatments, etc. Current HITs are vendor or IT professional designed systems, which are often inconsistent with the data collection needs of the clinicians[13]. Moreover, the deployed HITs are often inflexible in that it is either impossible or time-consuming to evolve them according to clinicians' current and changing needs [6, 13]. Consequently, this leads to either unintended consequences[7] such as creation of more work for the clinicians[18], or HIT failures[14]. A better solution is to design flexible HITs that would allow the clinicians to easily and quickly modify the system based on their needs[13]. In this work, we focus on a specific implementation of HIT, the electronic health record(EHR)[9, 19], and propose a flexible EHR (fEHR) system. The fEHR system provides an opportunity to clinicians to build new templates/forms over existing EHR systems based on their needs while maintaining the high-quality of the database. We conduct a usability study in a nurse-managed health services center to assess whether clinicians can use the proposed system while countering the inconsistency and inefficiency problems posed by current HITs.

The fEHR system is flexible in that it allows clinicians to easily extend the data collection functionality based on their customized needs. This is made possible by incorporating a form-based approach; the system leverages the facts that the knowledge of forms is already ingrained in clinicians and the semantics of forms could guide the generation of a database. Clinicians express their data collection needs by designing data-entry forms using the graphical user interface provided by the system. The system analyzes the clinician-designed forms and builds a database over the existing EHR system. The technical significance of fEHR is that the generated database satisfies the desirable properties of a high-quality database and is thus comparable to a professionally

designed system. We theoretically prove that the derived database satisfies the normalization property with respect to the given needs, and informally describe the fulfillment of other properties such as completeness, correctness, and compactness, with respect to storing the collected data.

The ultimate goal of this research is to make the HITs more effective and efficient in the healthcare domain, and to facilitate the work of clinicians rather than impose more learning burden on them. For this, we conduct a usability study with 5 clinicians working in a nurse-managed health services center. We introduce the clinicians with the fEHR system and observe them as they specify a variety of data collection needs by designing forms using the fEHR interface. Considering the current challenges and work pressures of clinicians, we measure their performance at various levels. We observe an overall improvement in their efficiency, confidence, and understanding, in using the system. Also, we find that, in 19 out of 20 cases, the clinicians are able to model and build need-based forms with 100% percent accuracy within a short span of time.

We compare the proposed system with earlier works that present generic solutions for non-technical users to build databases. Most of these works use a form-based approach[3, 23, 5, 1, 22] but none ensures the generation of a high-quality database. Ignoring quality has many consequences such as logical inconsistencies and update anomalies. Through this work, we overcome these shortcomings and make the following two contributions. The fEHR employs an algorithm that uses the hierarchical semantics of the form and the classic principles of database design to generate a database. It thus ensures that **the clinician-built database satisfies all desirable properties of a high-quality database**. We evaluate the usability offered by the fEHR system against the real experiences of clinicians. The high performance of the clinicians, in terms of efficiency and effectiveness, suggests that the system has **the potential to negate the inefficiencies and inaccuracies caused by current inflexible HIT designs**. The consistent improvement in clinicians' efficiency, confidence, and understanding, in using the system, suggests that the system provides them with **a conducive environment for learning to specify and model needs**.

The remainder of the paper is organized in the following order. Section 2 describes the architecture and functionality of the fEHR system. Section 2.1 presents the key approach underlying the system. Sections 2.2-2.4 describe the three components of fEHR system that interplay to generate a high-quality database. Section 3 presents the user study results and findings. Section 4 presents a short review of literature on self-help database design tools and Section 5 concludes the paper.

2. THE FLEXIBLE EHR SYSTEM

The fEHR system has been designed to enable the clinicians to extend the underlying database based on their data collection needs. To provide flexibility, the system follows a form-based approach in that it leverages the clinicians' high familiarity quotient on forms, and the rich information embedded in forms to guide database design. Figure 1 shows the fEHR architecture and an example of a need being translated to a database. The system has 3 components: (i) fInterface that allows clinicians to self-design need-based forms, (ii) tGeneration that creates an equivalent tree structure for a form, (iii) dAlgorithm that explores the tree structure to build a high quality database.

2.1 Form-based Approach

The fEHR system is form-based in two ways. First, it allows the clinicians to express their needs through forms and provides them a flexible way to extend the system easily and quickly. Second, it uses the information extracted from forms to guide the generation of high-quality databases reflecting the needs of the clinicians.

For several decades, clinicians have been using forms for data collection. In this paper, we explore a new usage of forms, wherein clinicians model their data collection needs as data-entry forms. We observe that the structures of data collection needs and data-entry forms are very compatible. For instance, if a clinician has a *simple* need of gathering patient's information such as name, age, address, and primary care physician's name, the form in Figure 1 would accurately capture this need. If the clinician has a more *advanced* need of gathering information related to a patient's health such as heart rate, blood pressure (systolic and diastolic), etc., then the form in Figure 2 would accurately reflect this need. We classify a need as simple or advanced based on the following assumptions. The form features corresponding to a simple need would be easy to understand for a clinician and the features corresponding to an advanced need would be more difficult to understand. Forms make a powerful communication medium for clinicians for need specification because forms are easy-to-understand commonplace objects [8], and the knowledge of the organization scheme of forms is encoded within humans [12]. With the ability to self-design forms, clinicians achieve the flexibility to specify simple as well as advanced needs based on their own perception of data. Eventually, the generated database closely reflects the clinician's model as much as possible[13].

Another advantage of having a form-based approach is that forms could be analyzed to retrieve important information about databases[8]. In our approach, we first derive a tree structure out of a form, and then analyze and translate it into appropriate database elements. Moreover, our study confirms that forms not just help in designing a database, but also in designing a high-quality database [21]. Next subsections describe various modules of the fEHR system.

2.2 fInterface

The fInterface is the form design interface of the system that provides clinicians the flexibility to extend the system based on their needs. In order to facilitate quick and easy specification of needs, this graphical user interface has been kept simple in terms of terminology as well as design. To attain a simple terminology, the interface concepts are represented through regular and intuitive terms. The interface allows clinicians (users) to specify a title for the form, add category, and its fields and also specify the format (textbox, radiobutton, dropdown list, etc.) for each field. Since data-entry forms have a hierarchical structure, we allow a category to contain subcategories with sub-fields as shown in Figure 2 where *Smoking* is a subcategory (contained in the category *Current Problems*) with subfields *Do you smoke* and *If yes, how ....* The interface allows a sequential flow of user steps. The form in Figure 1, representing a simple need, can be designed using the following steps (excluding button clicks):

1. Enter the title, *Patient Information*

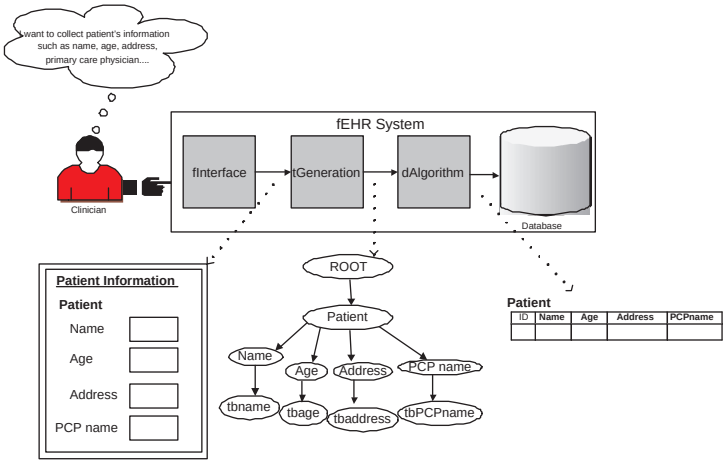

Figure 1: The Flexible Electronic Health Record(fEHR) System

2. Enter the category, *Patient*.
3. Enter the field, *Name*.
4. Enter the format, textbox for the previous field.
5. Enter the field, *Age* (See Figure 3).
6. and so on ...

At each step, the user is presented with a limited number of *concept gateways* associated with certain form concepts. This restricts users to a limited number of possible next steps and thus helps minimize design errors such as specifying a field without any format. Then, based on the gateway entered by the user, a component is added to the form being designed.

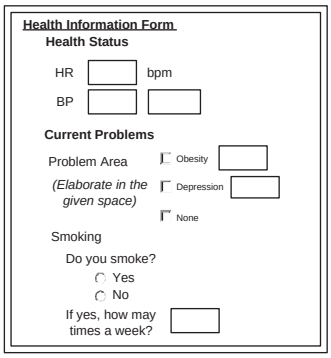

Figure 2: A form representing an advanced need. *Elaborate in the ...* is a Supporting Text, *Obesity* is an extended checkbox option, *bpm* is a unit, *BP* is a multiformat field, *Do you smoke* and *How many times ...* make a condition

Our next concern was simplicity in design. Simplicity has been considered the first usability principle in context of designing interfaces for the HITs [15]. The idea is to have a minimalistic interface that keeps only those features that are relevant to clinicians, and thus prevent the clinicians from being overwhelmed with feature overload. We analyzed 51 data-entry forms currently being used in healthcare. The dataset was collected from the Web and consisted of forms in various formats such as *HTML*, *pdf*, and *doc*. Table 1 summarizes the list of features found in the dataset. Given the success of these healthcare forms, we believed that clinicians would already be familiar with the features offered by this dataset. The fInterface emulates these features and hence remains under the boundaries of clinician-friendliness. The interface of our first prototype, fEHRv1, supports all the frequently occurring features (no. 1-6) found in the dataset. It allows the clinicians to specify up to one level of subcategories within a category as the dataset had at most one level of nesting between categories and subcategories.

Table 1: Feature Summary of 51 Healthcare Forms

	Feature	Frequency
1	Text Labels (Categories, Subcategories, and Fields)	60.12
2	Text Inputs (textbox, textarea)	24.23
3	Radiobutton Groups	8.61
4	Checkbox Groups	8.59
5	Drop down lists	6.82
6	Supporting Texts, Units (Fig. 2)	2.70
7	Multi-formats (Fig. 2)	0.39
8	Ext. Checkbox/Radiobutton (Fig. 2)	0.11

2.3 tGeneration

The goal of the fEHR system is to induce the data collection needs, captured in a clinician-designed form, into the database. It may appear very straightforward at first, and a naive approach is to translate the form into a single wide relation with form fields as the attributes of this relation, e.g., the form in Figure 4 would be translated into a relation called InjectionInformation with Time, Site, GivenBy, ReactionSize, ReviewedBy as the attributes. However, there are two problems related with the approach. First, the hierarchical information is lost. As a result, the grouping information, e.g., *Reaction By* and *Reviewed By* semantically belong to the same group *Reaction*, is not captured. Second, intuitively, this database is not normalized as we have placed heterogeneous attributes (such as Time and ReactionSize)

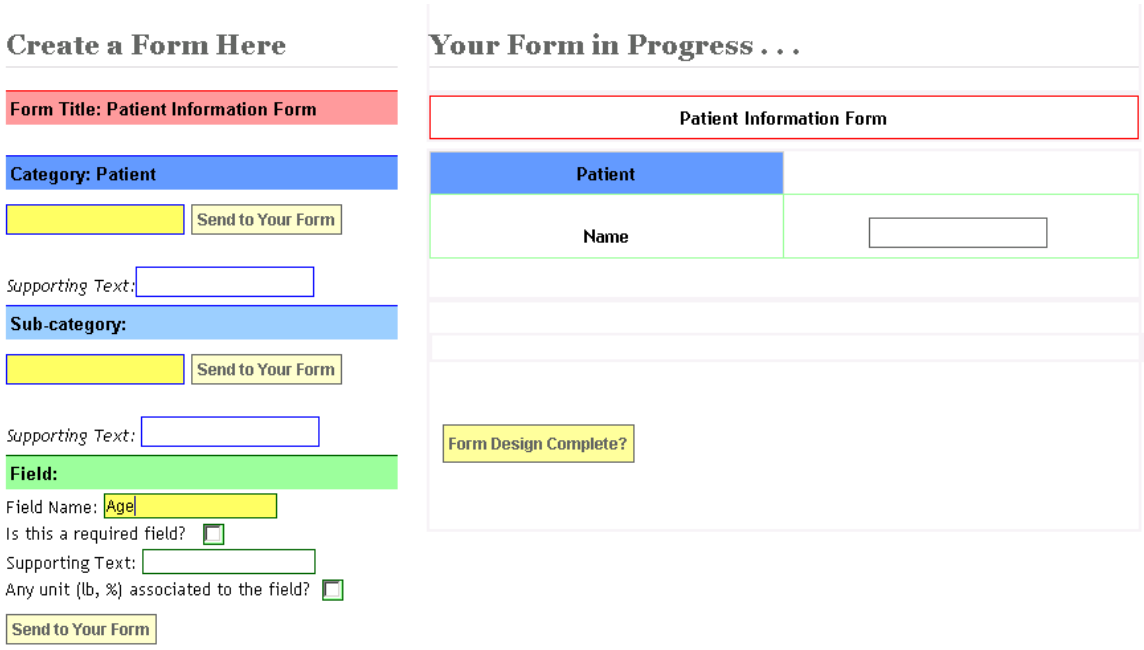

Figure 3: A screen-shot of the fInterface at step 5 (Section 2.2). The left division is a placeholder for the clinician to enter various form components. There are 3 concept gateways in this case: Category, Sub-category, and Field; and the clinician decides to enter the Field gateway. The right division shows the form being designed.

under the same relation. To generate a high quality database suitable for real world usage, it is thus important to accurately capture the hierarchical semantics of a form. We propose a hierarchical construct, *form tree*, to represent a form and the function of the tGeneration module is to derive a form tree out of a form. A form tree is a conventional tree where each node, except the root, represents a distinct form component. A form component is either a text label(category, field, etc.) or a form input (textbox, checkbox, etc.). We now formally define a form tree.

DEFINITION 2.1 (FORM TREE). *A Form Tree is defined as a labeled, directed and ordered tree, $\mathcal{FT} = (N, E, root)$, where*

- *N is a finite set of nodes. Each $n \in N$ has a label l and a type t. Each node is of one of the following types: element node, representing a text label on the form; format node {textarea, textbox, radiobutton, checkbox, dropdown}; or value node. The function $\lambda(n)$ returns the node label, and the function $\tau(n)$ returns the node type.*
- *E is a finite set of edges, such that for an edge $(n_i, n_j) \in E$, $n_i, n_j \in N$, n_i is called parent and n_j is called child.*
- *$root \in N$ is the root node of the tree.*

Figure 4 shows a form and its corresponding form tree. The relationships among the form components are maintained through parent-child(category-field, category-subcat., field-format) or sibling(category-category, field-field) associations in the tree. Some previous studies [10] have also proposed a tree representation for a form. The form tree, used in our work, differs in that it contains the format nodes corresponding to the form inputs such as *tbtime*, *tbsite*, etc. We argue and shortly show that these nodes are equally important in generation of a high-quality database. It should be noted that the tGeneration module dynamically derives a form tree based on the user actions captured in the tool. While this module currently only processes the fInterface-designed forms, it can be extended to process elsewhere-designed forms by combining some existing form information extraction algorithms[16].

2.4 dAlgorithm

We now present the dAlgorithm of fEHR that converts a form tree into a high-quality database while accurately escalating the associations among various tree nodes into the database. In the current fEHR system, we consider using a traditional relational database for storing and managing data. A (relational) database $\mathcal{D} = (I, R, \Sigma)$ is a 3-tuple, where I is a set of relations, R is the schemas of the relations, and Σ is a set of integrity constraints imposed on the relations (see the tables on the right-hand side of Figure 4). A relation consists of a set of tuples conforming to its schema. The schema for a relation specifies the name of the relation, the name of each column (or attribute or field), and the type of each column. The integrity constraints impose conditions that the tuples in relations must satisfy. Here, we consider the *key* and *foreign key* constraints. A key in a relation is a subset of the attributes of the relation that uniquely identifies a tuple. A foreign key in a relation T is a set of columns F that *references* the key of another table T', and imposes a constraint that the projection of T on F is a subset of the projection of T' on the key of T'. We

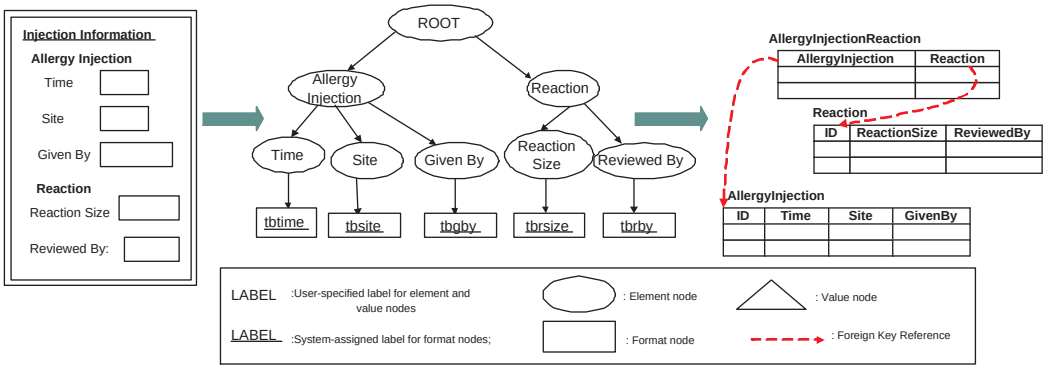

Figure 4: fEHR Example: Translation of a Simple Need

use the notation $T(A_1, A_2, ..., A_n)$ to represent the schema of a relation T with attributes $A_1, A_2, ..., A_n$, and we use the notation $T.A_i$ to refer to the attribute A_i of the relation T.

We aim to generate a high-quality database with "good" properties. Standard database textbooks (e.g., [11]) contain rich content about designing *normalized* databases with "good" properties including avoiding logical inconsistency and update anomaly. In addition, we are interested in several other properties such as *correctness*, *completeness*, and *compactness*. In this section, we first informally describe the satisfaction of *correctness*, *completeness*, and *compactness* properties, and then formally present the *normalization* property with respect to form tree.

Informally, the *correctness* property indicates that after a form tree is converted into a database, the semantic relationships among the elements of a form tree should be correctly reflected in the database. The *completeness* property stipulates that every data item collected on the form is stored in a place in the database. And the *compactness* property requires that each concept should be uniquely represented in the database. The properties guide our translation process from a form tree to a database. The following 2 examples illustrate how the algorithm generates a database which is correct, complete and compact in terms of storing data collected on a form.

Let us consider deriving a database corresponding to a simple need shown in the form in Figure 4. The preliminary inputs to the recursive dAlgorithm are the root of the form tree and an empty database. The algorithm works in the following manner ignoring the format nodes at the moment. Starting with the tree root, a relation T is created for each internal non-root node n. To every created relation T corresponding to n, attributes are added such that the parent-child associations between n are its child nodes are recorded correctly in the database. The procedure is repeated if a child node n_i is an internal node, otherwise an attribute corresponding to n_i is added to the relation T, e.g., Time is an attribute of AllergyInjection. Finally, the relationship among all the child nodes of the root node, i.e., among all the categories in the given form, is recorded by creating a relationship table between any pair of child nodes, e.g., the relationship between the two relations Reaction and AlleryInjection is captured by the join relation AllergyInjectionReaction.

Let us now consider deriving a database for an advanced need as shown in the form in Figure 5. The methodology is similar to the one devised for simple needs with an exception that here we also process the format nodes. Consider the internal format nodes, *rbvis* and *cbsym*. The radiobutton node *rbvis* indicates the presence of multiple exclusive options in the form which should get translated to values in the database as shown in the relation Visit. The checkbox node *cbsym* indicates the presence of multiple non-exclusive options in the form which should get translated to distinct (boolean) attributes as shown in the relation Symptoms. Furthermore, this form also has a subcategory BP under the category *Current State*. This relationship is captured by creating a join table CurrentStateBP between the respective relations to cover all possibilities of cardinalities between the two entities. Algorithms 1 and 2 present the details of the birthing process. The resultant database is correct, complete, and compact.

We now consider the normalization property, i.e., the traditional criteria for avoiding certain undesirable characteristics. A normalized database (i.e., in the Third Normal Form [11]) is defined with respect to a set of functional dependencies. Since in our case a user only provides data entry forms, we need to automatically deduce functional dependencies from data entry forms and translate the functional dependencies to the associated database. We represent the translation relationship between a form tree and a database as a set of correspondences. Specifically, given a form tree $\mathcal{FT} = (N, E, root)$ and a database $\mathcal{D} = (I, R, \Sigma)$, a correspondence $\mathcal{FT}$:P/e $\rightsquigarrow$ $\mathcal{D}$:D.d relates a node $e \in N$ of the form tree reached by a simple path P to an element d in the database component D. A simple path P is always relative to the root of the form tree, in which "/" is used to represent a parent/child relationship. A database component could be either a tuple in a table T or the schema of the table. We say a functional dependency $\mathcal{D}$:D.d$_i \rightarrow$ $\mathcal{D}$:D.d$_j$ in the database is *associated with* the $\mathcal{FT}$ if there are two correspondences $\mathcal{FT}$:P/e$_i$ $\rightsquigarrow$ $\mathcal{D}$:D.d$_i$ and $\mathcal{FT}$:P/e$_j$ $\rightsquigarrow$ $\mathcal{D}$:D.d$_j$, where P/e$_i \rightarrow$ P/e$_j$ is a functional dependency in the $\mathcal{FT}$.

To formally specify the normalization property, we consider the integrity constraints derivable from a form tree. Element nodes may represent either entities or attributes of entities in the application domain described by the form. Format nodes and value nodes in a form tree are directly related to attributes or values of entities. A parent-child edge between two element nodes gives rise to a functional dependency relationship between the nodes. Figure 6 illustrates various cases of deriving integrity constraints from a

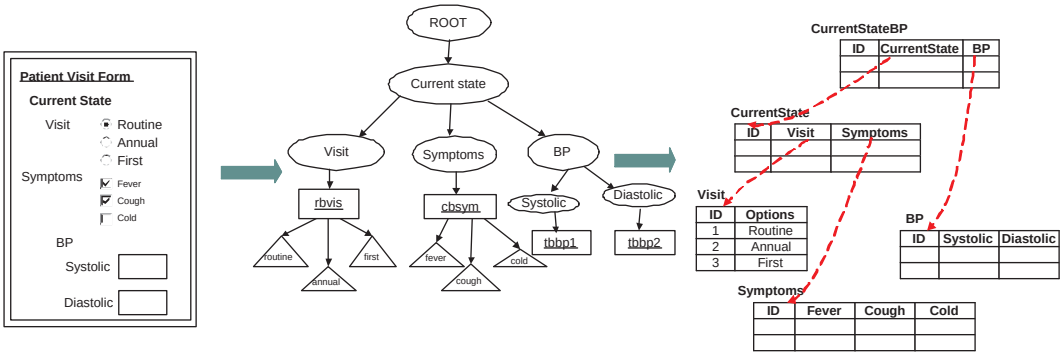

Figure 5: fEHR Example: Translation of an Advanced Need

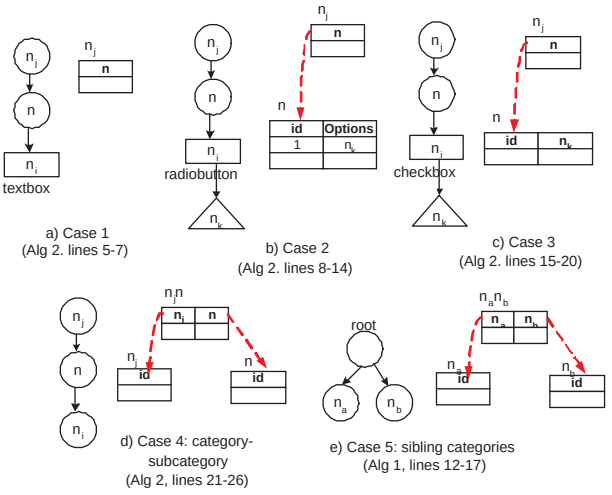

Figure 6: Cases for Database Birthing

form tree. The following theorem states the normalization property of a database derived from a form tree.

THEOREM 2.1. *Let $\mathcal{FT}$ be a given form tree and $\mathcal{D}$ be an empty database. The database $\mathcal{D}$ evolved by the procedure* dAlgorithm*$(\mathcal{FT}.root, \mathcal{D})$ is normalized with respect to the form tree $\mathcal{FT}$.*

3. USABILITY EVALUATION

In the previous section, we described the methodology for developing the fEHR system that provides the clinicians with the flexibility to extend an existing system based on their needs while ensuring the high-quality of the database. In this section, we present a report on the real experiences of clinicians while using the fEHR system. To demonstrate that the clinicians are able to use the system to extend an existing database, we conducted a user study. Considering the problems posed by the rigid design of HITs, we studied the following aspects of usability: (i) whether clinicians can use the system **efficiently** in terms of time duration (to counter the inefficiency cased by current HITs), (ii), whether they can specify their needs **effectively** (to counter the inconsistency between their needs and databases), (iii) whether they can **understand** the system easily and can use it **confidently** (to keep their workload within limits).

Algorithm 1 dAlgorithm

Input: A node $n \in \mathcal{FT}.N$, the database $\mathcal{D}$ (obtained from previous recursion).
Output: A database $\mathcal{D}$
Steps:

1: **if** $n \neq \mathcal{FT}.root$ **then**
2: **if** n_i is not a textbox, $(n, n_i) \in \mathcal{FT}.E$ **then**
3: Create a new relation T named $\lambda(n)$.
4: **end if**
5: **end if**
6: **for** each node $n_i \in \mathcal{FT}.N$, $(n, n_i) \in E$ **do**
7: $\mathcal{D}$:= processChildNode$(n_i, \mathcal{D})$ /* *See Algorithm 2* */
8: **if** n_i is an element node **then**
9: $\mathcal{D}$:= dAlgorithm$(n_i, \mathcal{D})$ /* *Recursion* */
10: **end if**
11: **end for**
12: **if** $n = \mathcal{FT}.root$ **then**
13: **for** each distinct pair of nodes n_a, n_b, such that $(n, n_a), (n, n_b) \in \mathcal{FT}.E$ **do**
14: Create a relationship table T_r named $\lambda(n_a)+\lambda(n_b)$.
15: Add foreign key references from T_r to T_a and T_b.
16: **end for**
17: **end if**
18: **return** $(\mathcal{D})$;

The study was conducted with 5 clinicians working in a nurse-managed health services center. All participants had 1-4 years of work experience and had diverse job titles (health educator, fitness trainer, behavioral health consultant). Out of all the participants, 3 (P1, P2, P3) were comfortable and 2 (P4, P5) were only moderately comfortable working with computers. While all had some experience with paper forms, none had any knowledge of databases. We conducted the study in 2 rounds. First, the participants were asked to use the first prototype, fEHRv1, to specify certain simple and short needs to be incorporated in the underlying database. Then, the system was revised into the second prototype, fEHRv2, and a second round was conducted with advanced and longer needs. The system showed the potential to provide an efficient and effective edge to current HITs, and was easy to learn for the clinicians.

3.1 Study Round 1: Simple Needs

During the first round, the participants were given a short training on the following fInterface concepts: title, category,

Algorithm 2 processChildNode

Input: A node $n_i \in \mathcal{FT}.N$, the database $\mathcal{D}$ (obtained from the previous recursion)
Output: A database $\mathcal{D}$
Steps:

1: Let n be the parent of n_i, and T be the relation corresponding to n.
2: **if** $n \neq \mathcal{FT}.root$ **then**
3: Let n_j be the parent of n and T_j be relation corresponding to n_j.
4: **end if**
5: **if** n_i is a textbox **then**
6: Add a new attribute named $\lambda(n)$ in relation T_j.
7: **end if**
8: **if** n_i is a radiobutton or dropdown field node **then**
9: Add an attribute named "Options" to T.
10: **for** each node $n_k \in \mathcal{FT}.N, (n_i, n_k) \in \mathcal{FT}.E$ **do**
11: Insert a new record in T such that value of *Options* attribute is $\lambda(n_k)$.
12: **end for**
13: Create a foreign key reference from T_j to T.
14: **end if**
15: **if** n_i is a checkbox field node **then**
16: **for** each node $n_k \in \mathcal{FT}.N, (n_i, n_k) \in \mathcal{FT}.E$ **do**
17: Add an (boolean) attribute named $\lambda(n_k)$ to T.
18: **end for**
19: Create a foreign key reference from T_j to T.
20: **end if**
21: **if** n_i is an element node and $n_j \neq null$ **then**
22: **if** $n_j \neq \mathcal{FT}.root$ **then**
23: Create a relationship table T_r named $\lambda(n_j) + \lambda(n)$.
24: Add foreign key references from T_r to T_j and T.
25: **end if**
26: **end if**
27: **return** $(\mathcal{D})$;

sub-category, field, format (radiobutton, checkbox, textbox, textarea, and dropdown). We first wanted to assess the form building abilities of the participants, and hence, devised the *build* task wherein they were asked to replicate a given paper form template on the system. Next, we wanted to assess whether they could model their needs into forms, and design those forms using the system. For this, we devised a *model-and-build* task wherein they were asked to create a form based on a given word description of data collection needs.

We devised 5 different pairs of build and model-and-build tasks for the participants. The average **form scale** was 17, i.e., the forms associated with the tasks required on an average 17 steps to be created using the fInterface. The needs considered in this round were simple and small-scale. During the tasks, participants were provided with any assistance if needed; and at the end of the tasks, they were asked to fill up a usability survey about the interface. To measure the performance of participants, we devised two metrics: **duration ratio**, obtained by dividing the task completion time (in minutes) by the form scale; and **assistance ratio**, obtained by dividing the number of assistances required by a participant by the form scale. Each task generated 4.2 relations, 5.8 non-key attributes, 1.8 values, and 3.2 foreign key references on an average. Following were the key observations of this round.

- As shown in Figure 7a, the duration ratio remained under reasonable limits (0.05-0.56) and the total duration ranged from 1 to 9 minutes. The duration ratio was only a little higher for the model-and-build task which shows that the participants could quickly model their needs into forms. The participant P3 was an exception who asked several curiosity driven questions during the build task and hence took longer to finish it. The participant P4 did some initial preparation on paper before performing the tasks and hence took relatively longer than others.

- No participant committed any model/build error.

- The value of the assistance ratio dropped for most participants as shown in Figure 8a, indicating that they became more confident when using the system for the second time. Participant P5 was however an exception and required a little more help during the model-and-build task. During the latter task, participants took several fine modeling decisions independently, e.g, choosing between category and subcategory based on term semantics, and choosing between radiobutton and checkbox based on field semantics.

- Although all participants had an implicit knowledge of the form concepts, they needed help to distinguish among concepts like category, sub-category, and fields. All participants could thoroughly understand different kinds of formats easily which shows that participants re-used their knowledge of paper-forms.

We now present the second round of user study conducted with the revised prototype, fEHRv2.

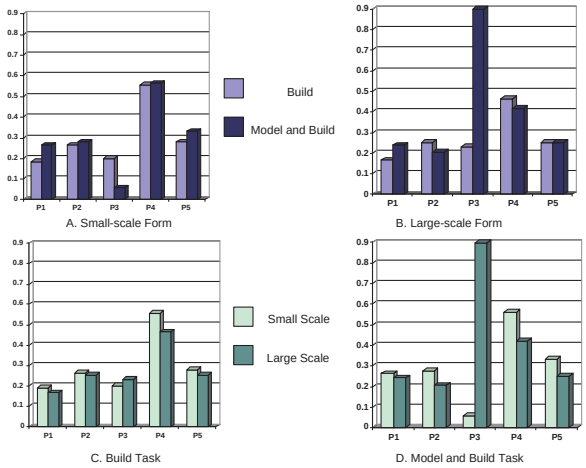

Figure 7: Duration Ratio

3.2 Study Round 2: Advanced Needs

The second round differs from the first one in that the participants were asked to use the system to specify advanced and large-scale needs. In the second prototype, fEHRv2, we added features mentioned in the survey results such as multi-formats, conditions, extended radiobuttons/checkboxes as well as other untested features such as unit and supporting texts (See Fig. 2). We made necessary revisions to the

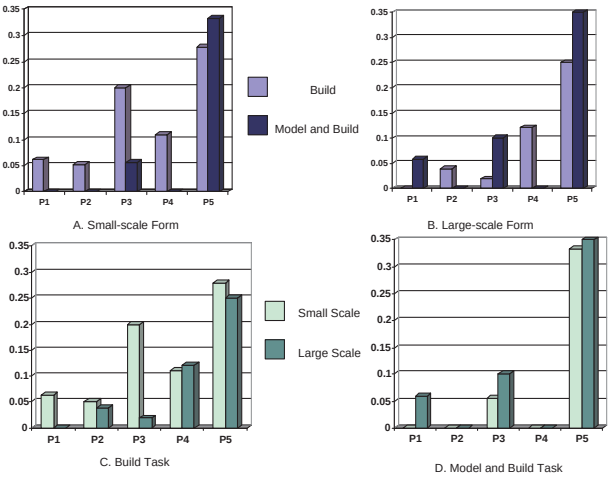

Figure 8: Assistance Ratio

tGeneration and the dAlgorithm modules but do not present them due to space constraints. The average form scale used for this round was 47.4 and the average complexity of the generated database was 6.2 relations, 13.8 attributes, 10.4 values, and 4.6 foreign key references. Following were the main observations of the second round.

- The duration ratio for large-scale forms ranged from 0.16 to 0.46 as shown in Figure 7b, and the time duration ranged from 7 to 19 minutes. Figures 7c and 7d illustrate that, in this round, the duration ratio decreased for both tasks even though the forms scaled up. This indicates that the participants could finish the tasks reasonably quickly even for the large-scaled forms and became more efficient in general. An exception was the participant P3 who considered at least 3 design alternatives during the model-and-build task and hence took longer. However, the duration ratio followed no fixed pattern for large-scale forms in terms of both tasks as seen in Figure 7b.

- There was one building error committed by a participant who skipped a component during the former task, and there were no modeling errors.

- The assistance ratio dropped for the build task (Figure 8c) even though the participants transitioned from shorter and simple to longer and advanced needs. However, the ratio increased for the model-and-build task (Figure 8d) indicating that the participants required more assistance while modeling advanced and longer needs into forms. Between build and model-and-build tasks for large-scale forms, the assistance ratio followed no fixed pattern as seen in Figure 8b.

- The understanding of form concepts improved across all participants. While modeling, the participants further applied their knowledge of formats, and began making conscious choices, e.g., between radiobutton and dropdown list, a participant preferred dropdown list as it would save the form space, while another preferred radiobutton as it works better for the users of healthcare systems. Out of the newly introduced features, condition and supporting text were the most understood and multiformat was the least understood one.

3.3 Discussion

The usability study helped us in synthesizing several findings. In this section, we present two main findings of the study; related to the potential of fEHR system in reducing current problems, and related to the ease with which clinicians could learn using the system; and discuss the limitations of the system and the study.

3.3.1 Key Finding 1: System Potential

To find the potential of the fEHR system in dealing with current challenges, we measure the effectiveness and efficiency of participants in using the system. Effectiveness is measured in terms of how accurately a participant models a need into a form and builds the form into the system. In all but one case, participants finished the tasks with 100% effectiveness. The unsuccessful case was because of a building error committed by a participant who skipped a component while building forms. Also, the participants showed great efficiency in using the system that ranged from 1 to 9 minutes for simple small-scale needs, and 7 to 19 minutes for advanced large-scale needs. The only exception was a participant who considered several design alternatives while modeling an advanced need and hence took relatively longer than the rest. In sum, the fEHR system shows clear signs to counter the current HIT-related challenges of inefficiency and impedance mismatch between needs and databases.

3.3.2 Key Finding 2: System Adoption

To make sure the system does not create more learning work for the participants, we measure the improvements in their efficiency, confidence, and understanding in using the system. The efficiency of participants consistently improved on moving from small-scale to large-scale needs even with a little practice of using the system. We assess a participant's confidence by measuring the assistance ratio for a given task. All participants were very confident in using the system for specifying small-scale needs for both the tasks. For the build task, there was an apparent improvement in confidence even though the participants transitioned from simple and smaller to advanced and longer needs. For the model-and-build task, the participants, however, required more assistance for modeling longer needs. The understanding of form concepts improved greatly during the second round for all participants and they even started synthesizing their knowledge of form concepts and domain knowledge to consider different design alternatives. Thus, we conclude that the system is easy to adopt and provides a conducive learning environment to clinicians in terms of modeling and building need-based forms.

3.3.3 Limitations

While using the fEHR system, the participants mainly faced two challenges, which directly affected their performance. First, they faced difficulty in learning the form terms. Even though the terms chosen were commonplace, it was difficult to put the terms in alignment with their data collection needs. Second, they mostly got stuck and sought assistance while taking modeling decisions during the model-and-build task. These decisions were both fine, e.g., choice of

format, and coarse, e.g., the ones affecting the overall structure of forms, involving the choice of category or subcategory and leading to multiple design alternatives. While our training helped in overcoming the main challenges, a much deeper solution is needed to suit the preferences and constraints of the clinicians. Future HITs should provide such assistance and design recommendations to the clinicians to further improve their performance.

While interacting with the participants and observing them, we found two kinds of learners among them: visual and digital. The visual learner (P3, P4, P5) has a categorical style of thinking and understands concepts better when visualized on paper. The digital learner (P1, P2) has a linear style of thinking and understands concepts better mentally while working with the system. The learning style had nothing to do with their computer proficiency. Our sample was limited to 5 participants; with larger sample, more style and challenges can be uncovered. Customizing the system or the training method for each learning style would be greatly beneficial in future.

4. RELATED WORKS

Health informatics research has shown the need for supporting user-designed information tools [13]. The work in [13] studies 6 clinicians in a trauma hospital, observing how they use a self-designed tool, a *clipboard*, for communication and decision making. The clipboard contains paper cut-outs of important data assembled from multiple sources. Based on the usage patterns and the reliance on this tool, it is concluded that future HITs should facilitate the clinicians to self-design tools according to their working needs.

While the above-mentioned study[13] is specific to healthcare, the inclination toward developing flexible tools is apparent from several generic works that enable non-technical users to design databases. IIS*Case[22, 20] is a tool that automatically generates a database schema based on user requirements. Users model their needs into a *form type*, a tree structure with *component types* as its nodes. Each component type has a name, and a set of attributes with associated domains and constraints. By specifying a form type, the user indirectly specifies the relations, attributes, and constraints of the back-end database. Microsoft's Deklarit [2] is a model-driven tool for application development and is designed for users with no background in databases and modeling. Users are required to specify *business components* containing attributes, data types and key constraints. A business component may contain another component giving rise to a hierarchical unnormalized structure. The relationships among components are inferred using universal relationship naming convention and a database schema with key and referential integrity constraints is created accordingly. ZohoCreator [5] is a tool that allows users to design forms and hence databases. Using ZohoCreator, a user can build applications by designing form templates. Each template is a flat set of attributes. Each attribute can have one of the multiple available formats such as single line, multi line, email, date, checkbox, etc. FormAssembly[3] is another successful commercial tool for data-driven application design. It enables users to design data-entry forms, and automatically generates a schema corresponding to the form. While designing a form, the user is asked to provide the questions and their respective formats (textbox, checkbox, etc.) to appear on the form. These questions could be arbitrarily grouped under sections and subsections. JotForm[1] and PerfectForms[4] are some other form-based tools.

Appforge[23] provides an application building tool to non-technical users. It employs an algorithm that translates the user-specified actions into a sophisticated ER model with multi-way relationships and aggregations. The user works with concepts like role, page, view, form, and container. The user creates forms; and the tool translates each form to a separate entity. The user can also design views (or nested views) and add new columns to existing views using a schema navigation menu. This helps in creating relationships and relationship attributes among the entities of the back-end model. The system was tested on a group of 6 users, including 2 researchers who were database experts with advanced degrees in computer science, 2 researchers who were non-database experts with advanced degrees in computer science, 1 managerial position holder trained in computer science, and 1 recruiter familiar with using database applications. Appforge was easily understood and used by the users with advanced computer science degrees. The other 2 users, however, faced understanding and usage difficulties. The interface was hence re-designed based on the difficulties and confusion faced by them.

The above discussion shows the substantial progress made in terms of facilitating users to design need-based applications. However, these solutions cannot be directly applied to healthcare because of the associated usability and quality issues. While Appforge[23] is the only work to report the usability results, it has only been tested on experienced computer users. Moreover, Appforge exposes the underlying schema to the user (e.g., schema navigation menu), which is likely to impose additional cognitive burden on users as the database scales up. Also, certain approaches [2, 22] expect users to understand technical terms such as business components, attributes, constraints, form type, domains, etc. These approaches are hence less likely to be adopted by the clinicians who have no background in computers and databases. Other form-based tools like FormAssembly [3] and Zohocreator[5] make use of simple real-world concepts and show signs of high usability. However, they do not report their algorithm of database generation and hence the quality of the resultant database cannot be guaranteed. The proposed fEHR system differs from the existing works in that, it is usable and learnable as confirmed by the user study, and it is technically sound and ensures a high-quality database comparable to a professionally designed system.

5. CONCLUSIONS

This work is motivated by the vision of the US government to effectively induce HITs into healthcare by 2015 [15], and by the challenges faced by the clinicians while working with the rigidly designed HITs. To counter these challenges, we have proposed a flexible system, fEHR, using which clinicians can easily and quickly extend an existing EHR system as per their needs. The fEHR system employs a form-based approach wherein it takes as input the clinician-designed form corresponding to a given need, and translates the form into a database. We formally present that the employed algorithm generates a normalized database with respect to the clinician's needs, and informally describe the satisfaction of other database quality properties such as correctness, completeness, and compactness. With this, our first contribution is that the proposed system guarantees that

the clinician-built databases are of high-quality comparable to any professionally developed database. To ensure that the system can directly help in improving the condition of HITs in healthcare industry, we conducted a case study with clinicians working in a nurse-managed health services center. The clinicians could perform the given tasks of modeling and building forms with 100% accuracy in all but one case. They could use the system for designing the databases based on short and simple as well as long and advanced needs within a span of few minutes in most cases. Also, there were signs of improvement in clinicians' levels of efficiency, confidence, and understanding in using the system. Therefore, our second contribution is that the system shows the potential to reduce the current problems of HITs, particularly, the inefficiency faced by clinicians, and the inconsistency between clinician's needs and databases. In addition to this, the system is adoptive in that it helps the clinicians to learn and improve their need modeling and form building skills.

The user study helped in identifying several future directions for improving the system. Considering the main challenges faced by participants, we intend to re-design fEHR's interface such that it helps clinicians in taking modeling decisions and suggests design alternatives to them based on controlled medical vocabularies. The case study participants suggested addition of new features like calculated fields, table widgets, etc. While addition of such advanced features is technically possible, what is challenging is to introduce them without imposing any learning burden on the clinicians. We also intend to extend the fEHR design such that new forms could be integrated with an existing database while maintaining all desired properties in the resultant database and while managing other unforeseen implications[17].

6. ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for providing valuable suggestions for revising this paper. This research work is supported in part by NSF Career Grant IIS 0448023, and NSF Grants CCF 0905291 and IIP 0934197. The user study was approved from the Institutional Review Board at Drexel University.

7. REFERENCES

[1] Jotform - easiest form builder. http://www.jotform.com/.

[2] The model-driven tool for microsoft visual studio 2008. http://www.deklarit.com/portal/hgxpp001.aspx?12.

[3] Online web forms, surveys & questionnaires - formassembly.com. http://www3.formassembly.com/.

[4] Perfect forms, a smarter way to manage and monitor your business. http://www.perfectforms.com/.

[5] Zoho creator. https://creator.zoho.com/.

[6] Y. An et al. Collaborative social modeling for designing a patient wellness tracking system in a nurse-managed health care center. In *DESRIST '09*, pages 1–14, Philadelphia, PA, USA, 2009. ACM.

[7] J. S. Ash, M. Berg, and E. Coiera. Some unintended consequences of information technology in health care: The nature of patient care information system-related errors. *JAMIA*, 11(2):110–112, 2004.

[8] J. Choobineh, M. V. Mannino, J. F. Nunamaker, Jr., and B. R. Konsynski. An expert database design system based on analysis of forms. *IEEE Trans. Softw. Eng.*, 14(2):242–253, 1988.

[9] C. M. DesRoches, E. G. Campbell, S. R. Rao, K. Donelan, T. G. Ferris, A. Jha, R. Kaushal, D. E. Levy, S. Rosenbaum, A. E. Shields, and D. Blumenthal. Electronic Health Records in Ambulatory Care – A National Survey of Physicians. *N Engl J Med*, 359(1):50–60, 2008.

[10] E. C. Dragut, T. Kabisch, C. T. Yu, and U. Leser. A hierarchical approach to model web query interfaces for web source integration. *PVLDB*, 2(1):325–336, 2009.

[11] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems, 3rd Ed.* Addison-Wesley, 2000.

[12] D. W. Embley. Nfql: the natural forms query language. *ACM Trans. Database Syst.*, 14(2):168–211, 1989.

[13] A. P. Gurses, Y. Xiao, and P. Hu. User-designed information tools to support communication and care coordination in a trauma hospital. *J. of Biomedical Informatics*, 42(4):667–677, 2009.

[14] M. I. Harrison, R. Koppel, and S. Bar-Lev. Unintended consequences of information technologies in health care -an interactive sociotechnical analysis. *Journal of the American Medical Informatics Association*, 14(5):542–549, 2007.

[15] S. Karmakar. How to design a usable and meaningful emr application. Technical report, eMids Technologies, 2010.

[16] R. Khare, Y. An, and I.-Y. Song. Understanding deep web search interfaces: A survey. *SIGMOD Record*, 39(1):33–40, 2010.

[17] R. Koppel and D. Kreda. Health care information technology vendors' "hold harmless" clause: Implications for patients and clinicians. *Journal of the American Medical Association*, 301(12):1276–1278, 2009.

[18] T. Lee. Nurses' experiences using a nursing information system: early stage of technology implementation. *Computers, Informatics, Nursing*, 25(5), 2007.

[19] J. A. Linder, J. Ma, D. W. Bates, B. Middleton, and R. S. Stafford. Electronic health record use and the quality of ambulatory care in the united states. *Archives of Internal Medicine*, 167(13):1400–1405, 2007.

[20] I. Luković, P. Mogin, J. Pavićević, and S. Ristić. An approach to developing complex database schemas using form types. *Softw. Pract. Exper.*, 37(15):1621–1656, 2007.

[21] M. V. Mannino and J. Choobineh. Research on form driven database design and global view design. *IEEE Database Eng. Bull.*, 7(4):58–63, 1984.

[22] J. Pavicevic, I. Lukovic, P. Mogin, and M. Govedarica. Information system design and prototyping using form types. In *ICSOFT (2)*, pages 157–160, 2006.

[23] F. Yang, N. Gupta, C. Botev, E. F. Churchill, G. Levchenko, and J. Shanmugasundaram. Wysiwyg development of data driven web applications. *Proc. VLDB Endow.*, 1(1):163–175, 2008.