Updating Database Schemas Without Breaking the UI: Modeling Using Cognitive Semantic Categories

Evangelos Kapros, Simon McGinnes
School of Computer Science and Statistics
The University of Dublin, Trinity College
Dublin 2, Ireland
{evangelos.kapros, simon.mcginnes}@scss.tcd.ie

ABSTRACT

Data management user interfaces are ubiquitous in information systems and web-based applications. From the oldest spreadsheet to the most modern database, end users and administrators alike have interacted with tabular data. Usually, each concept is represented by a *table* and *columns*. Change to the structure of each concept requires structural change to the tables and columns, which is costly. Tailor-made database and web applications may overcome this obstacle by designing UIs on top of the data layer, providing some degree of data independence. However, changes in their schemas do not automatically propagate into the user interface, and so their maintenance is expensive.

In this paper we present a user interface that lets the end user alter the schema without the need for programming skills, eliminating the need for expensive software maintenance. To this end we propose an automatically generated user interface to include schema and data management functions. We built and evaluated an Adaptive Information System user interface (AIS UI), incorporating schema evolution functionality. In usability testing, first-time users were able to perform various data management tasks equally fast or faster than users using Microsoft Access, and on average ~43% faster than users using Microsoft Excel. Task completion rates using the AIS significantly exceeded those using Microsoft Access and were comparable (>95%) with those using Microsoft Excel.

Author Keywords

Adaptive information systems; databases; spreadsheets.

EICS'14, June 17 - 20 2014, Rome, Italy

ACM 978-1-4503-2725-1/14/06…$15.00.
http://dx.doi.org/10.1145/2607023.2607027

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces–*Graphical user interfaces (GUI)*. H.2.m. Database Management: Miscellaneous.

INTRODUCTION

Data management applications are a main production force in office environments. Many SMEs, NGOs, and other small organisations depend on so-called "productivity applications" such as Microsoft Access databases or Google Docs spreadsheets.

These organisations often lack the luxury of the "buy or build" choice, since their resources may be too limited to build bespoke data management solutions or to purchase expensive enterprise software applications. Instead they use spreadsheet (SPR) and database (DB) software to create their own ad hoc data management systems.

However, relying on systems of this type can be problematic. One reason for this is that the systems need to be maintained and enhanced over time. For example, change can occur due to disagreement over semantics (the so called "Tower of Babel" problem), miscommunication, and change in business needs.

In many cases the maintenance and enhancement of a system requires amendment to its underlying data schema (which may be either explicit, as in a relational database, or implicit, as in a spreadsheet). Changing the schema of a database or spreadsheet makes it necessary to re-write queries or formulas and to re-design the user interface, to reflect the changes in the schema. This can be costly, since expert IT skills are required (typically from the IT departments of these small organisations or external contractors). However, IT departments and external vendors are not always best-placed to understand user needs, and are not necessarily trained in user-centered design methods, so the results are not necessarily ideal.

In this paper, we propose that the end users themselves should be able to adjust the schemas of their own data management applications. To this end, we have defined an adaptive data model for data management, by including concepts (tables) and attributes (columns) in extended "soft" schemas, so that end users can manage entities and

relationships without breaking the UI. Operations on the schema are performed using the familiar desktop metaphor.

We implemented this idea in an "Adaptive Information System" UI (AIS UI), a web application that allows the manipulation of arbitrary databases and database structures using a desktop-like GUI. Our AIS UI is a tool for editing and managing a database that allows the user to work primarily with a graphical representation of the schema; in this sense, it is important to note that the adaptivity in our approach lies predominantly on the Information Systems end of the system rather than just the UI. We evaluated our approach by conducting a laboratory-based user study with 30 participants. Each participant used either our system, database software (Microsoft Access), or spreadsheet software (Microsoft Excel). We recorded the performance time and the completion rate of the participants performing a scenario of eight tasks, and we used a System Usability Scale (SUS) questionnaire to evaluate the relative usability of the three types of software.

RELATED WORK

Spreadsheets

Research has shown that most organisations rely on spreadsheets for their data management [3, 12]. There are several reasons for this, including a lack of usable end-user systems. End users have been reported to "shun enterprise solutions" [12] and 70% use spreadsheets frequently or occasionally, most commonly for "sorting and database facilities" [3].

However, spreadsheets are notoriously error-prone; because they lack critical database management functionality, serious issues of data integrity can arise. There are real risks in the widespread dependence on spreadsheets [16]. In response, research has looked at ways of achieving database-like functionality in spreadsheets, such as the management of relationships [1]. However, the underlying issue of schema evolution in the spreadsheet context has received little attention. Work on "semantic spreadsheets" has improved modelling capabilities in spreadsheets, but maintains the problematic distinction between the roles of authoring and use [15, 10].

Relational Databases

Codd proposed the relational model as a way to translate concepts into data structures. Although his original work provided the theoretical tools for a versioning system of tables and relationships [17], this system has not reached production. One can imagine that Codd's original grasp of relational databases successfully addressed schema evolution issues, however there is no implementation of his original model; even Codd himself later abandoned the idea.

Relational databases have offered little flexibility for end users, who had to depend on professional database administrators to deploy any change. Research in Schema Evolution focuses on this very problem of adapting a database schema to changes. This research field has shown that changes in schemas represent a significant cost to organizations. In [4] changes in the database schema are reported to affect up to 70% of queries, which have to be manually reconfigured. Some theoretical models to address this problem have been constructed, but real systems incorporating schema evolution functionality are hard to find [5].

NoSQL Databases

NoSQL is an umbrella term for databases that reject some or all of the constraints of the relational model. Such databases include document databases and graph databases. These technologies allow fast querying of large volumes of data, and they can accommodate schema evolution more smoothly than relational databases. However, the data in a NoSQL database must be communicated to the end user via a user interface. At present it is not clear how the schema change would be accommodated by user interfaces in a non-discontinuous manner and without the need for expert IT skills.

User Interfaces for Data Management

User interfaces for relational databases typically use tabular views for data, which may or may not be programmable. The following convention is followed: data are viewed in tabular form; relationships are displayed in a different screen and might use lines to illustrate how tables are linked; database administrators and end users have separate views; end users can have custom-designed interfaces which have to be amended once the tables or the relationships change. A notable novel approach of querying databases can be found at [24] and even though querying is beyond the scope of this paper it is an approach worth mentioning as it brings visual interface affordances to data management.

Applications such as HyperCard and FileMaker have facilitated metaphors similar to the desktop for data management. However, they also suffer from the "Tower of Babel" problem, as each application developer has to build their own user interface, as views are not embedded in the authoring environment.

Automatically-Generated Interfaces

Our goal is to allow end users to change the schema of an application without having to reprogram its user interface. This means that the interface must be re-generated in some way. Work has been done on automatically-generated interfaces exists and addresses various issues. For example, some early theoretical work on interfaces which can be adapted to various devices is ability-based [7]. Other work has facilitated adaptation to end users' capabilities (personalisation) [23] or users' tasks [24]. The majority of this work has provided various formalisations of user-interface languages to facilitate adaptation (e.g. [4, 23, 24]). Research on [23] is of particular interest with regard to

letting end-users themselves retrieve information at a semantic web setting.

Moreover, work on ontology evolution has given useful results on change in semantics while using an automatically generated interface [13, 14]. These results use annotated ontologies and, in many case, the interfaces are generated semi-automatically [5]. Similarly, work on dynamic data management has given useful results [15,16,17] but has not, in general, addressed user-interface or usability issues.

Other relevant research has used cognitive semantics to create non-domain-specific top-level ontologies that can be used as a basis for arbitrary schemas [9]. This idea is explored in the next section.

THE ADAPTIVE INFORMATION SYSTEM

We approach the problem of schema evolution by turning conceptual models (schemas) into data rather than, for example, hard-coded table structures, and by allowing "lazy" transformation of existing data following schema change. To test these ideas, we built and evaluated a prototype Adaptive Information System (AIS) (see Figure 1 and Figure 4). Current application design practice requires conceptual models to be hardcoded into software structures (classes, windows, tables, etc.) Our AIS avoids this practice. Instead, the AIS is constructed from generic, domain-independent structures and reusable functionality. The model-as-data is termed a soft schema; in our prototype it is stored as XML, although any logically-equivalent way of storing data would suffice. The soft schema is read and interpreted by the AIS at run-time (see Figure 2 and Figure 3). The soft schema is a properly normalised relational data model, with some additions, but it is stored as data rather than being hardcoded in application structure.

To provide domain-specific functionality, yet also exhibit conceptual data independence, the AIS meets several conditions. First, it reacts at run-time to a soft schema, providing a user interface which looks and behaves similarly to those of conventional domain-specific applications. To date we do not provide specialised behaviour for different types of data by responding to semantic categories embedded in the soft schema, but this is planned. Currently we present all data in tabular form (Figure 1).

The AIS can store and retrieve data corresponding to multiple soft schemas with guaranteed data integrity. The AIS has no advance knowledge of the schemas it will be used with, and how they may change. The data corresponding to each soft schema must be able to co-exist and be used with data stored for other soft schemas, regardless of their structures. Therefore we store data in a domain-independent way, but retain intact the conceptual structure for each instance of data. Our prototype meets that requirement by storing the data using XML and using XML tags to denote structure. But, again, any logically-equivalent storage mechanism would suffice. For example, another prototype uses a graph database to represent the same information in a natural way but with the potential for improved runtime performance.

Archetypal categories

The AIS provides domain-specific behaviour by responding to the currently active soft schema. Each concept (entity type) in the soft schema represents something that data can be stored about. Therefore, the AIS provides CRUD (create, read, update, delete) functionality in respect of every concept in the schema. Design heuristics are applied automatically to produce a "reasonably usable" interface directly from the conceptual model. The principle has been applied and tested in a number of web and client-server application environments [18]. Dialog design takes into account general rules of interaction and layout, as well as responding specifically to the data types used for attributes in the soft schema, the relationships between concepts, and so on.

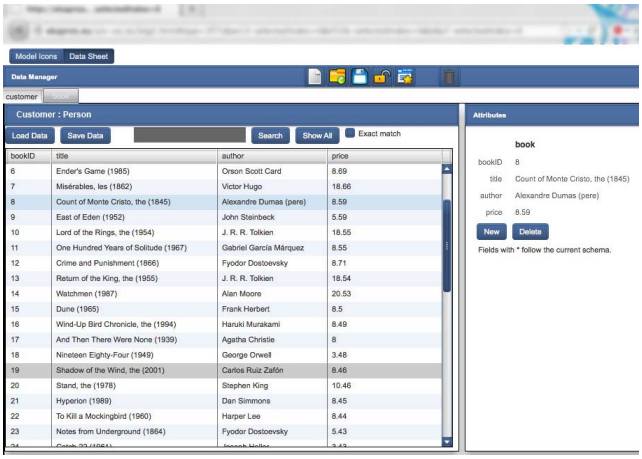


Figure 1. The Data Manager tab of the AIS UI.

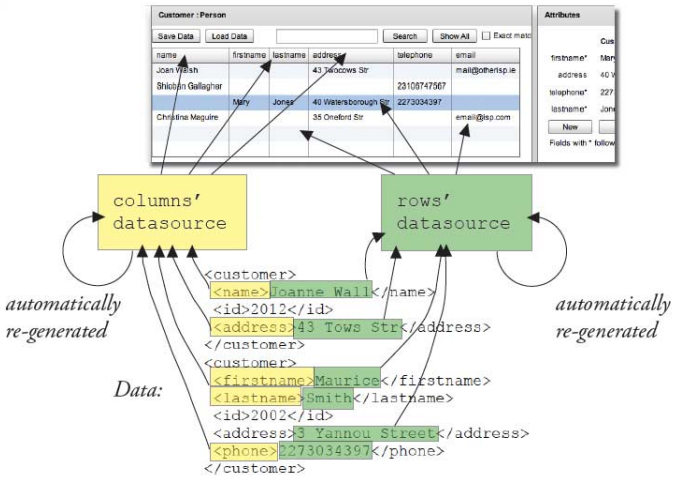


Figure 2. The concepts of the soft schema form the columns of the data grid, while the attributes its cell contents.

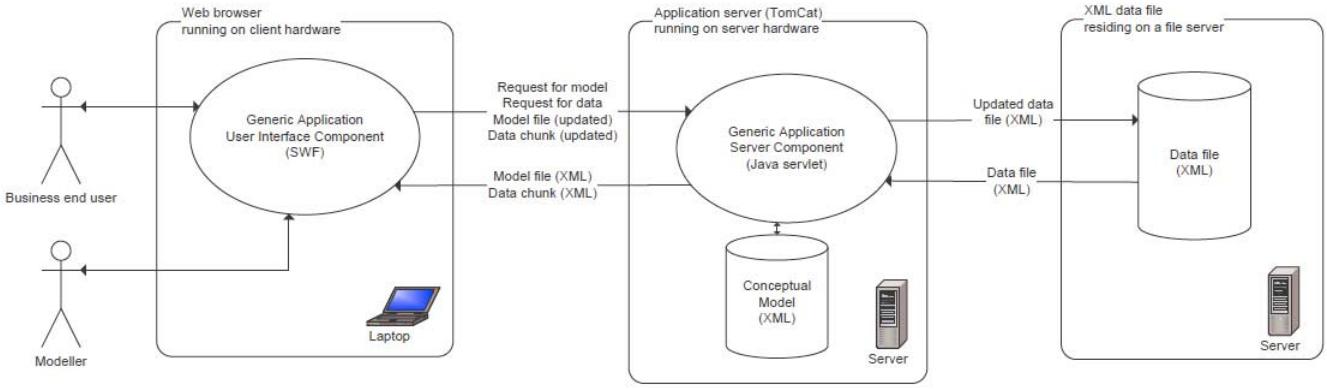


Figure 3. Architecture of the AIS UI.

We therefore sought to embed data and attributes into soft schemas, so that it could be used automatically by an AIS to render more domain-specific behaviour. It is achieved by linking each concept in the soft schema with a particular archetypal category (major cognitive semantic category [19,20,21]). The prototype AIS uses nine archetypal categories: *people, organisations, places, documents, activities, physical objects, conceptual objects, systems and categories* [18]. Using archetypal categories presents advantages during modelling; for example, it allows aspects of models to be predicted, helping to speed up modelling and reduce error. This makes end-user modelling easier, opening up the prospect that end users could use a suitable AIS to provide application functionality they want without recourse to IT specialists [22]. That is, these categories serve as a "top-level ontology" thus facilitating the flexibility, interoperability, and integrity of software applications.

End users can manipulate the model using the icons at the desktop-like representation of the soft schema (see Figure 4) and manipulate data and attributes using the tabular view of Figure 1. Using the archetypal categories to generate the UI means that the latter does not break when the end users make changes to the schema.

Relationships between concepts are accommodated implicitly. The end user can simply add a concept as an attribute of another concept, thus creating an implicit relationship. For example, a user might use the concepts tab of Figure 4 to locate the concept *purchase*. Then, they might go to the Attributes panel on the right hand side and click "New…" to add a new attribute, so as to define what is a purchase. There, they could add an attribute *purchaseDate* of the predefined *date*, or they could add a *location* which, as can be seen at the Concepts tab, is of category *places*. Thus, the end user has implicitly made the relationship "a purchase has-a location".

An Example: The "Bookplace" Database

As an illustration of the AIS in use, let us consider a simple example. A bookstore called "Bookplace" has stores in five locations, and each customer purchases one or more books.

The Bookplace manager can use the AIS UI to create icons in the model manager for this schema: they can make a new icon for the store location, which they can label as a place, an icon for the purchases, which they can label as activities, another for customers (people), etc.

Once the end user has made a *customer* concept, which falls under the category *people*, they can click on the icon to bring up the customer tab in the attributes panel on the right-hand side of the screen. There, they can make new attributes by clicking "New", choosing from a menu what type of attribute they want (e.g., text), and naming their attribute (e.g., first name).

Thus, the attributes panel represents a "has-a" relationship (i.e., the customer *has a* first name). The same mechanism is used to create joins; we can add an icon of another concept to the window to represent the fact that the two concepts are related (e.g., a purchase has a location). This is similar to the desktop metaphor in the sense that the desktop is (often) agnostic about the meaning of each icon: a folder might include an icon of a document or an icon of a folder or of a software application.

Double clicking the customer icon will bring up the data manager tab. There, the user will be able to click on "new" to add new customers, or open a file with existing customers. Moreover, the user can delete, edit, or search for a customer on this tab.

Adding and opening the contents of a concept follows usual desktop conventions: clicking on an icon operates on the icon itself; double clicking opens the content related to this icon.

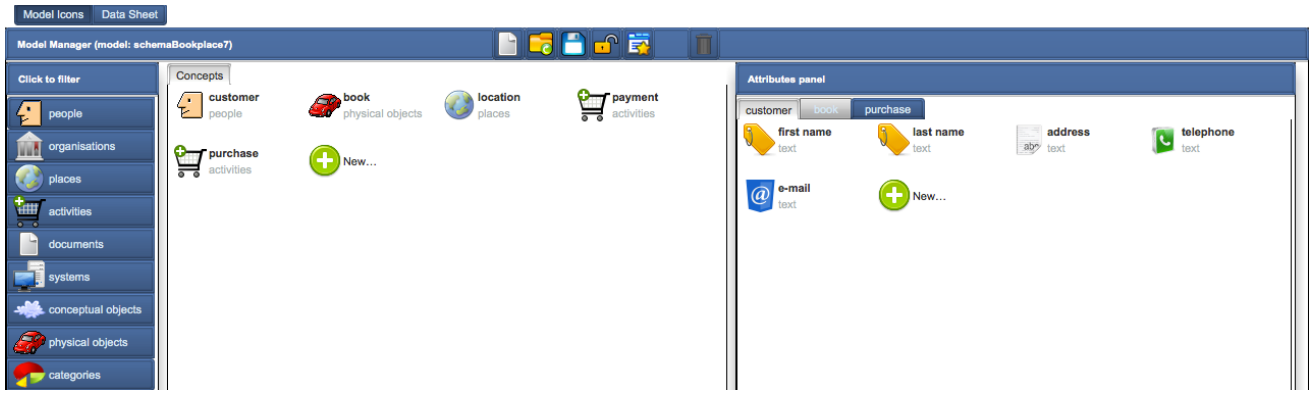


Figure 4. The Model Manager tab of the AIS UI.

Managing schema changes is as easy as managing desktop icons. Assuming that the manager of the "Bookplace" want to add a new payment method, they would only have to select the icon for *purchase* and add a new attribute icon, then go to the data tab and change the data accordingly. Note that this change will require no middleware/UI reprogramming. Data migration is optional; it can be done immediately or at a later time, or not at all.

Now, let us assume that the solution of modelling the relationship "purchase has-a purchase method" by representing purchase method as an attribute proves to be insufficient, so the Bookplace manager makes a decision to represent purchase methods as a new concept. Then, the end user can make a new concept *purchase method*, and the existing functionality will not break. Then, they can relate the new concept to the concept *purchase*, and (if they want to) migrate their data to this new structure.

In addition, users can right click an item to edit its properties; that is, they can set the cardinality of a relationship to allow-one or allow-many, or make an attribute mandatory. So far, relationships are modelled implicitly: allowing-many *book* items in a *purchase* will model a 1:N relationship. If a user would want to model an N:M relationship they would have to allow-many *purchase* at the *book* window. In the future we plan to provide a different menu in a drop-down fashion to accommodate this functionality.

USER EVALUATION

The evaluation of the AIS UI included a usability study in a laboratory environment where users conducted a set of tasks. The users consisted of thirty individuals in the 25-35 age range. All were students in the information systems/computer science area. Most had at least two years' work experience.

The experimental procedure was as follows. Each user was given a simple model of a bookstore, implemented in either AIS UI, Microsoft Access (DB), or Microsoft Excel (SPR). The allocation of software to user was made at random. Each user carried out a scenario of eight tasks, as follows.

1. Add new data of various types.
2. Add new attributes, e.g., a new payment method.
3. Make changes to the model, e.g., each purchase will change so as to have many product types.
4. Search for a particular piece of data.
5. Add and rename a concept.
6. Edit and delete a piece of data.
7. Handle missing relationships, e.g., enter a purchase where a location is missing.
8. Handle missing attributes, e.g., enter a payment where the payment method is missing.

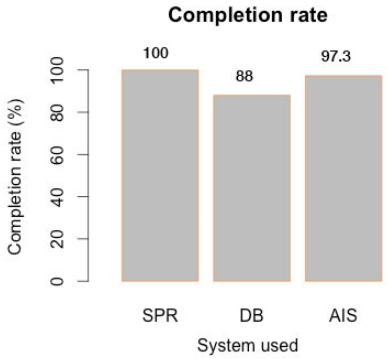
The starting schema was constructed by the authors in Excel, then exported to XML, and finally imported to Access and the AIS UI in order to create the three instances of the database to be evaluated.

The users loaded the schema and then followed instructions on a website given to them which had a detailed description of the tasks. Beside each task were two fields for the users to fill in: the time it took them to perform a task, and a checkbox to report if they actually finished the task successfully. No explicit talk-aloud protocol was enforced, although some participants voluntarily made comments.

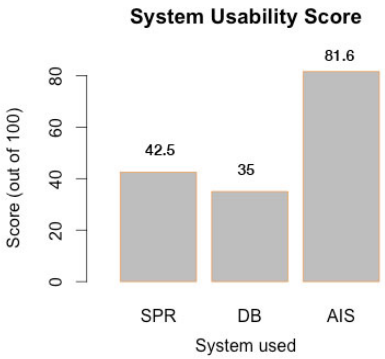
After completing the tasks, the users were asked to fill in a System Usability Scale (SUS) questionnaire [26].

Addressing Bias

The experiment was designed to address potential sources of bias in the following ways.



a)



b)

Figure 5. a) Completion rate for the eight tasks per system, b) reported usability score using SUS.

Ideally, the experiment would have taken place in a real working environment. However, a laboratory setting was used due to limitations in resources, and thus attention was paid to recruiting subjects who would represent the target users as much as possible. The users were professionals who were attending an evening information systems course, and came from a non-expert, non-computer-science background. While they might have occasionally had some computer related topics while studying, e.g., economics or mathematics, they were not programmers but typical users of productivity software at their workplace. Thus, we approached our target user, who is the non-expert office professional.

The experimenters supervised the procedure to make sure that the users were filling in the actual time employed. Moreover, the experimenters collected the final .xls, .mdb, and .xml files (from MS Excel, MS Access, and the AIS UI respectively) after the users had finished the experiment to make sure that the tasks, which have been reported as completed successfully, were indeed correct. However, the users were briefed that all tasks were voluntary and could leave the experiment at any time, so 11 of them either chose not to finish the scenario, or finished the scenario but didn't complete the SUS questionnaire.

Concerning the selection of the tasks, these were based on previous work [3] so as to form a realistic scenario according to what has been reported to be a common set of tasks among spreadsheet users. The intention for this choice was to fit the experiment to the common tasks and not vice versa.

Undoubtedly, one source of bias which was unavoidable was that all users were completely new to the AIS UI but had used Excel and Access before. The users were given a short brief concerning the experiment, but the results might still be potentially biased in favour of Access and Excel.

RESULTS

Analysis

Out of the 30 recruited users, 19 finished the scenario *and* filled in SUS questionnaires. We analysed the 19 entries that had both a finished scenario and a SUS questionnaire submitted.

The completion rate was 97.3% for the AIS UI, 100% for Excel, and 88% for Access.

	Estimate (minutes)	Std. Error	t-value	P-value
AIS	18.500	3.652	5.066	0.000115***
DB	24.714	4.977	1.249	0.229741
SPR	31.333	5.165	2.485	0.024403*

Table 1: Linear regression results for the user completion time per system.

	Estimate (minutes)	Std. Error	t-value	P-value
AIS	17.833	4.257	4.189	0.000413***
DB	23.333	6.020	0.914	0.371323
SPR	31.333	6.020	2.242	0.035862*

Table 2: Linear regression results for the total task time per user.

To accommodate statistical significance analysis, given the fact that there were missing data due to varying completion rates, we considered two cases: one where we interpolated performance time values for uncompleted tasks, and a more conservative case, where we set the time of each completed task to zero. In both cases normality (Q-Q) and homoscedasticity (Bartlett) tests showed the following.

Uncompleted task time interpolated.
The results passed the homoscedasticity and independence of error tests, so a linear regression analysis of the results was performed (Table 1). For the homoscedasticity hypothesis we have Bartlett's K-squared=1.0988, df=2, and p-value=0.5773, so the hypothesis holds (p-value>0.05).

Uncompleted task time set to zero.
These results also passed the homoscedasticity and independence of error tests, so a linear regression analysis of the results was performed (see Table 2). For the homoscedasticity hypothesis we have Bartlett's K-squared=3.3443, df=3, and p-value=0.3415, so the hypothesis holds (p-value>0.05). Also, we have lag=1, autocorrelation=-0.0605187, D-W statistic=2.04625, and p=value=0.612. The alternative hypothesis is: rho!=0. Since p-value>0 we do not reject the null hypothesis of rho=0. There is no autocorrelation and independence of the error hypothesis holds.

The SUS score for AIS UI was 81.6, for SPR it was 42.5, and for DB it was 36.

Discussion
The results show a statistically significant performance improvement when using the AIS UI instead of spreadsheets, up to 43% (Δ=13.500 minutes, p<0.05). In addition, the AIS UI users performed equally or better than database users (Δ=6.214 minutes, p=0.23) (see Figure 4).

In addition, completion rates for the AIS UI were similar to the ones of popular spreadsheet and database production software (88<97.3<100) (see Figure 5.a). More specifically, the completion rate of 97.3 for the given scenario means that only one user did not accomplish one task using the AIS UI.

These performance metrics are important and demonstrate some advantages of our system. However, it is the usability metrics that fully demonstrate the potential impact of this approach. Our system scored 81.6 points at the Systems Usability Scale, that is more than double than the database which scored 35 points, and almost double points than the 42.5 points of the spreadsheet (see Figure 5.b).

Where does this difference in usability lie? A qualitative observation might give us a hint. As noted before, a talk-aloud protocol was not enforced; however, two users who were using the AIS UI made useful comments, having been rather surprised by the interface they were using:

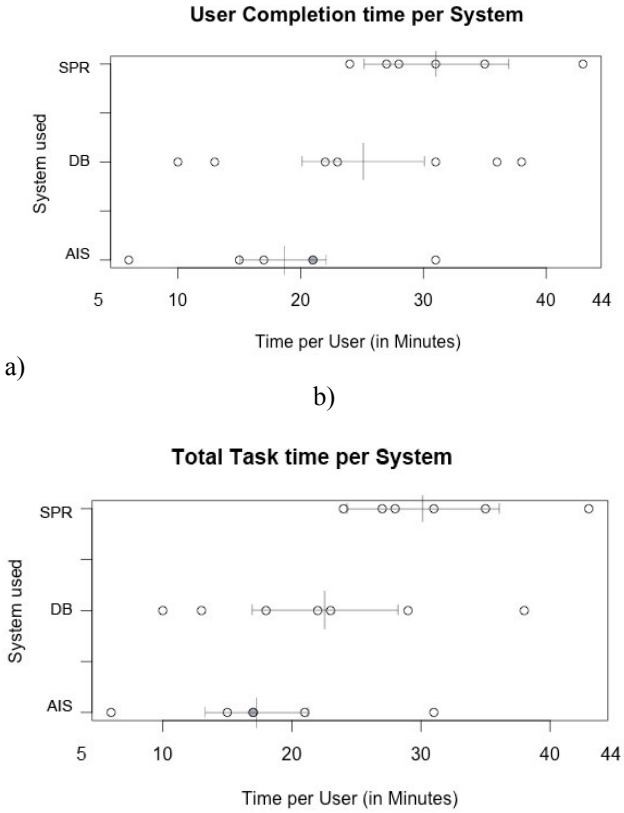


Figure 4: User performance per system. Each white circle represents one user; each gray circle represents two users. The mean is drawn as a vertical line, the standard error from the mean drawn as a horizontal bar. a) Scenario completion time per system, regardless of the completion rate for each system. b) Total task time per system (Uncompleted task duration = 0 min).

*"It makes perfect sense to use icons instead of columns, it's so more **usable** this way." (User 28)*

"I cannot believe that we've not been using this all this time—what were we thinking?" (User 30)

This supports the view that there is merit in extending the desktop metaphor to data management. Looking closely to the performance time of each task, we see that spreadsheet users performed well at direct data manipulation, i.e., tasks 1 and 5 (add, search average time=2.7 minutes) but not at managing relationships, i.e., tasks 3, 7, and 8 (average time=13.7 minutes). In contrast, database users performed better than spreadsheet users at managing relationships, especially at task 3 (avg.=5.2 min), but not better than AIS UI users (avg.=4.3 min). The average time to change the cardinality of a relationship was just 1.3 minutes for database users, 3 minutes for AIS UI users and 3.3 minutes for spreadsheet users (to find a working solution).

These results suggest that, in edge cases such as pure data manipulation and intensive schema change, the spreadsheet and the database respectively are fit for purpose (ignoring other potential concerns such as security, data integrity, system integration, etc.).

However, we propose that the AIS UI, with its familiar metaphor, offers a middle way and good performance for end users who need a fair bit of both schema and data management, where their resources do not allow them to invest in a bespoke solution or enterprise software.

The results show that it is beneficial indeed to manipulate schemas and data graphically, according to the motivation we presented at the introductory section. That is, large organisations that can afford experts on relational or document databases might not avail much of a desktop-like representation of schemas. Similarly, individuals who need to organise a single collection of data, e.g., personal finances, might be better off using a spreadsheet. Since the vast majority of organisations, SMEs, and NGOs do not lie in these categories, but need a middle level of data management complexity, it is probable that they will find a system like our AIS UI not only usable, but also useful.

FUTURE WORK

Future work includes building an AIS UI which is more scalable and which includes specific views for each archetypal category.

In addition, the scenario of this usability study demonstrated the usability of changing an existing schema but not of merging two or more similar schemas. Thus, further usability testing needs to be conducted. We plan to achieve this by putting the AIS UI into trial in small organisations to apply it to real-world situations instead of a laboratory experiment. We anticipate that evaluating with users in organisations onsite might provide a larger pool of users, too.

Moreover, we plan to implement a version based on graph databases. We anticipate that we will be able to improve relationship modelling, e.g., by explicitly joining attributes of concepts through a visual dropdown-like menu.

CONCLUSION

In this paper we presented the Adaptive Information System UI, which adapts and augments the desktop metaphor to facilitate data management. Our implementation allows direct manipulation of icons to perform operations on concepts, attributes and relationships. We evaluated our system by conducting a user study that compared the AIS UI with Microsoft Excel and Microsoft Access when used in a simple business data management scenario. The AIS UI performed equally well to or better than Access and significantly better than Excel (~43% on average), and its SUS usability score was almost double that of the scores of Access and Excel.

REFERENCES

1. Bakke, E., Karger, D.R., and Miller, R.C. A Spreadsheet-Based User Interface for Managing Plural Relationships in Structured Data. In *Proc. CHI 2011*, ACM Press (2011), 2541-2550.
2. Canfield Smith, D., Irby, C., Kimball, R., and Harslem, E. The Star User Interface, In *Proc. AFIPS NCC 1982*, AFIPS Press (1982), 515-528.
3. Chan, Y.E., and Storey, V.C. The use of spreadsheets in organizations: determinants and consequences. *Information & Management 31*, 3 (1996), 119-134.
4. England, D., Randles, M., and TalebBendiab, A. Runtime user interface design and adaptation. *Proc. BCS-HCI '09*, BCS (2009), 463-470.
5. Ertl, D., Kaindl, H., Arnautovic, E., Falb, J., and Popp, R. Generating high-level interaction models out of ontologies. *Proc. IUI SEMAIS'11*, CEUR (2011), 467-468.
6. Evamy, M. *World Without Words*. Laurence King Publishing, London, UK, 2003.
7. Gajos, K., and Weld, D.S. 2004. SUPPLE: automatically generating user interfaces. *Proc. IUI '04*, ACM Press (2004), 93-100.
8. Horton, W. *The Icon Book.* John Wiley & Sons, 1994, 13-16.
9. Kapros E., McGinnes S. Cognitive Semantic Categories as a Basis for Adaptive Information Systems. In *Semantic Models for Adaptive Interactive Systems*, Springer, London, UK, 2013, 43-57
10. Kohlhase, A., and Kohlhase, M. Spreadsheets with a semantic layer. *Electronic Communications of the EASST Volume X*, (2010).
11. McGinnes, S., and Amos, J. Accelerated Business Concept Modeling: Combining User Interface Design with Object Modeling. In *Object Modeling and User Interface Design: Designing Interactive Systems*, Addison-Wesley, Boston, MA, USA, 2001, 3-36.
12. Raden, N. *Shedding light on shadow IT: Is Excel running your business?* Hired Brains, Inc., Santa Fe, NM, USA, 2005.
13. Wach, E.P. Automated ontology evolution as a basis for adaptive interactive systems. *Proc. IUI SEMAIS'11*, CEUR (2011), 467-468.
14. Whitehouse, R. The uniqueness of individual perception. In *Information Design*. MIT Press, Cambridge, MA, USA, 2000, 103-129.
15. Zhao, C., Zhao, L., and Wang, H. A spreadsheet system based on data semantic object. *Proc. ICIME 2010*, IEEE (2010), 407-41.
16. Dahalin, Z., 2005. Risks of user-development application in small business. *International Journal of Management Studies (IJMS)*, 12(2), pp.41–52.

17. Codd, E. F. A relational model of data for large shared data banks. *Communications of the ACM, 13*(6) (1970), 377-387.
18. McGinnes, S.: Systems and Methods for Software Based on Business Concepts. 20050289524, *USPTO* (2005)
19. Moore, C.J., Price, C.J.: A Functional Neuroimaging Study of the Variables that Generate Category-Specific Object Processing Differences. *Brain 122*, 943-962 (1999)
20. Markman, A.B.: Similar and Different: The Differentiation of Basic-Level Categories. *Learning, Memory 23*, 54-70 (1997)
21. Caramazza, A., Mahon, B.Z.: The Organization of Conceptual Knowledge: the Evidence from Category-Specific Semantic Deficits. *Trends in Cognitive Sciences 7*, 354–361 (2003)
22. McGinnes, S.: Conceptual Modelling: A Psychological Perspective. Ph.D Thesis, Department of Information Systems, London School of Economics, University of London (2000)
23. Karger, D., Bakshi, K., Huynh, D., Quan, D., and Sinha, V. Haystack: A General Purpose Information Management Tool for End Users of Semistructured Data. *CIDR*, (2005), 13-26.
24. Borges, C., and Macías, J. 2010. Feasible database querying using a visual end-user approach. *Proc. EICS '10*. ACM, 187-192.
25. Puerta, A.R. A Model-Based Interface Development Environment**.** *IEEE Software 14*, 4, 1997, 41-47.
26. Brooke, J. SUS: a "quick and dirty" usability scale. In *Usability Evaluation in Industry*. London: Taylor and Francis. (1996).