The End-User Programming Challenge of Data Wrangling

Maria Gorinova
Computer Laboratory
University of Cambridge
m.gorinova@gmail.com

Karl Prince
Judge Business School
University of Cambridge
k.prince@jbs.cam.ac.uk

Sallyanne Meakins
Papworth Hospital
University of Cambridge
sallyanne.meakins@nhs.net

Alain Vuylsteke
Papworth Hospital
University of Cambridge
a.vuylsteke@nhs.net

Matthew Jones
Judge Business School
University of Cambridge
mrj10@cam.ac.uk

Alan Blackwell
Computer Laboratory
University of Cambridge
afb21@cam.ac.uk

Abstract
We present a case study of requirements for "data wrangling" capabilities in a healthcare application context. Data wrangling is an increasingly common requirement for data scientists, policy makers, market researchers, intelligence analysts, and other professions where existing data must be used in ways that were not envisioned when it was first collected. We characterise data wrangling as a programming problem, in which aggregate data must be restructured in ways that remain consistent with its semantic origins or ontological referents. We recommend the table as a lowest common denominator representational device, affording both direct manipulation and programming by example. We describe work in progress, in which we have identified new opportunities for clinical end-users to interact with the content of a customisable information system, through a focus on tables as an approachable analytic tool.

1. Introduction
Much of the work of the professional data scientist is concerned with 'data wrangling' – organising data into a form that will allow it to be used for statistical analyses or visualisations. Data journalist Simon Rogers, as with many data analytics consultancies and other less public-minded specialists, finds this the most time consuming and technically challenging aspect of the work, up to 80% of a typical data analysis project (Rogers 2013). In his guide for the budding data journalist, Rogers suggests that this might involve copy and pasting values between Excel tables, before converting to comparable units, getting rid of unnecessary columns, merging cells, and changing it into a format compatible with visualisation systems.

Much of this work is currently manual, although tools such as TextWrangler can automate the extraction of numerical data embedded within XML or other text formats, and mashup techniques such as Yahoo Pipes can be used to scrape data and aggregate data from websites. Recognising and exploiting regularity in textual data has been a topic of concern at PPIG in the past (e.g. Church 2008, Blackwell 2001). However, it is often the case that the data is already in a structured form, simply the wrong structure for the task at hand. The data of concern may be stored in database, or even simply embedded within a large table, requiring queries to extract it, joins to combine it, or 'pivot' operations to unpack and reorder a nested structure.

In this paper, we report early results from a project 'in the wild', in which data wrangling has been identified as a central issue for end-user professionals. Our collaboration involves researchers from a business school, a computer science department and an NHS hospital, with the goal of helping hospital clinicians to make better use of patient data. Our ultimate objective, as with other fields of data analytics and data journalism, is to create visualisations and conduct statistical investigations as an aid to improved policy and care. However, results from our first phase of fieldwork indicate that data wrangling is one of the most substantial technical obstacles to this goal. In the remainder of this paper, we describe the problem we have observed, discuss the specific technical opportunities that have been highlighted by our investigation of this domain, and present some initial design principles that we are now exploring as the basis for novel end-user programming tools appropriate to the users we have been meeting.

2. Background and previous research

2.1. Research context

The research that we report in this paper is part of a project entitled *Repurposing Clinical data for quality improvement in Critical Care* (ReCliC), funded by the Health Foundation. The goal of the ReCliC project, as expressed in its title, is to explore ways in which data can be used more effectively in clinical practice. We take a specific focus on the clinical context of critical (or intensive) care – a relatively complex healthcare context, involving greater degrees of monitoring and intervention than most healthcare work. One consequence of this complexity is that larger quantities of data are generated in a critical care context, and hence that automated approaches to working with that data are particularly likely to be profitable.

The customisable clinical information system that is the focus of the current research has previously been studied in the context of end-user programming research (Morrison & Blackwell 2009). In a previous study, we investigated the potential for end-user programming tools to offer additional value to such systems by hiring a professional programmer to carry out extension work, under the instruction of a clinical end-user (Blackwell & Morrison 2010). In that study, we observed that the most pressing need for system extension was with regard to structuring the data in new ways, allowing for aggregated reports to be generated. In this work, drawing on a wider variety of informants from additional institutions, we have again identified the problem of reorganising and restructuring aggregated data.

The current research focuses on five NHS Critical Care sites across the UK, all employing the same clinical information system (CIS). At each site we engaged with *key users* responsible for supporting the clinical information system as well as servicing requests for data. The roles of key users varied significantly as well as their range of technical and clinical skills. This included a nurse, pharmacist, consultants, IT staff and an information analyst. We also engaged with a wider group of *end users* who typically enter data into the clinical information system but also request data for various purposes. End users included both clinical and non-clinical staff.

The research methodology is based on conducting semi-structured interviews with research subjects as well as observations of how the clinical information system is used in routine practice. Additionally, we have attended user group meetings where challenges faced in using the clinical information system are discussed and users share learning. The qualitative data collected from these interactions are coded for relevant themes, discussed amongst the research team and refined through a repetition of these steps.

Future phases of this research will explore clinical potential for novel statistical analyses, including time series trends, sample comparisons, correlations and clustering. We expect that end-user programming perspectives will also be relevant in this future work. However, in the current study, we focus purely on the end-user challenge of gaining access to data, and organising it into a form from which statistical analyses might be conducted – the challenge of 'data wrangling'.

2.2. Previous research in data wrangling

Data wrangling tasks have previously been identified as a candidate for end-user programming techniques. For example, tools such as PADS (Fisher & Walker, 2011) and FlashExtract (Le & Gulwani, 2014) allow users to extract structured data from semi-structured data, by either describing with the aid of a description language how the data should look like (PADS) or demonstrating it through an example (FlashExtract). Other tools focus on improving ways of connecting content from several different sources into a single Mashup, through programming by example techniques (Tuchinda, Szekely, & Knoblock, 2008). Further to that Vegemite uses a spreadsheet-like interface, together with direct manipulation and programming by demonstration techniques to achieve that (Lin, Wong, Nichols, Cypher, & Lau, 2008).

The research strategy that we have chosen is to treat data wrangling as a *transformation* problem, in which one set of data must be transformed into another. Based on the observation by Rogers (2013) that much routine data wrangling is carried out in spreadsheets, and on recent observation of the value that tabular representations provide in end-user data analytics (Sarkar et al 2014), we focus on

approaches to transforming one table of data into another. Previous approaches include systems that use menus, demonstration or examples to specify a spreadsheet or table transformation program.

Menu-driven tools allow the user to create programs by specifying a sequence of operations through a menu. One such tool is AJAX (Galhardas, Florescu, Shasha, & Simon, 2000), which uses an SQL-like language and focuses on *data cleaning* – the process of resolving inconsistencies in the data, performing entity resolution and correcting errors. Other systems, such as OpenRefine (Verborgh & Wilde, 2013) allow some commands to be specified graphically, but mostly users need to write them in a command language. Potter's Wheel (Raman & Hellerstein, 2001), on the other hand, has purely menu-based interface, used by the end-user to describe a sequence of operations, which could then be saved and applied to new data in the future. It relies on users constructing transformations gradually, i.e. having a clear idea of what sequence of operations they have to execute. This implies a need for strong understanding of the semantics of the operations, rather than just their effect.

Wrangler (Kandel, Paepcke, Hellerstein, & Heer, 2011) and its improved version, Proactive Wrangler (Guo, Kandel, Hellerstein, & Heer, 2011), take the Potter's Wheel approach one step further, by extending the transformation language and introducing an interface that allows the user to demonstrate what their intent is. This starts by selecting columns or rows relevant to the transformation, and is followed by suggestions from the system regarding what the intended transformation might be. The suggestions given by the Wrangler and Proactive Wrangler systems address the problem of the user needing a clear idea of the operations they need to go through. However, neither of these systems manages to completely solve this problem, still requiring the user to understand the particular semantics of the backend transformation language.

FlashRelate (Barowy, Gulwani, Hart, & Zorn, 2015) and previously ProgFromEx (Harris & Gulwani, 2011) allow users to give examples of the spreadsheet transformations they want to perform. The transformations are then automatically synthesised. The advantage of this approach is that users only need to understand the effect of transformations, not their semantics. However, a resulting problem with these approaches is that if the user does not understand the semantics, they find it difficult to verify that they have created the right program, to correct possible errors in the synthesised program or understand how it might be reused in future.

Our aim is to ease data wrangling tasks by unifying existing approaches such as Wrangler and FlashRelate, prompting the user to specify an example of the effect of the desired transformation program, automatically synthesise a suitable program, and allow the user to understand and amend it when needed.

3. Preliminary findings

In this section, we discuss findings from our field research into the organisational factors that set the context for reuse of clinical data, and analyse the characteristics of the data itself from an end-user programming perspective.

3.1. Organisational factors in data reuse

Our initial field research identified two constituencies of people who might be regarded as ‘end-users’ from the psychology of programming perspective. Neither is formally trained in IT (apart from the product training provided by the CIS supplier, together with any local or user-group support). However, one group has acquired primary responsibility for the data wrangling task. Morrison and Blackwell (2010) presented a case study of one such individual, who was described in that publication as a ‘professional end-user.’ In the current analysis, we describe these individuals as ‘key users’. The other group wishes to make use of the data, but currently has no tools that they find suitable for extraction of data from the system. We call these (data) ‘end users’.

In the remainder of this section we describe the ways that these two groups interact with the system within their organisational context. There are also organisational constraints on their work, sometimes reflected in their formal posts. Key users are often people who have been assigned specific responsibility for system operation, while data end users are more often clinical staff or hospital managers who need access to data for other purposes. Some of these staff are paid for the time they spend analysing data or interacting with the CIS, while others engage with the system outside of their

primary duties. However, there is not a straightforward mapping from the category of paid professional duties to the key user / data end user distinction. In some sites, the key user is a clinician or manager whose primary responsibilities are in clinical treatment or hospital management rather than IT systems.

In studying the typical process of interaction, we found that data end users requested data for a variety of purposes, including regular reports for clinical and management purposes, data for audit purposes (often submitted for national audit requirements), and data for research purposes. Key users would agree any data request queries with end users, taking into account not only their understanding of what data was stored in the clinical information system but also how likely they were able to extract particular data out of the system. Investigating this process highlighted particular issues for both key users and end users.

The first stage we identified in servicing data requests is the negotiation between the key user and end user. Key users indicated that end users are often not very clear about what data they require and there is a need to elicit more specific information before the exact specifications are understood. End users had not often thought through queries and key users see part of their job as facilitating the development of the 'right' question. Part of this negotiation is for key users to make end users aware of what is pragmatically possible within the constraints of the clinical information system. Since key users know the CIS best, they have a sense of how queries need to be stated to be feasible and so need to negotiate end user expectations accordingly.

A significant issue highlighted by key users attempting to address requests for data is the limited functionality of the CIS in providing the means to extract data. A distinctive (and desirable) feature of the CIS is that the recorded data is fully customisable. Data fields and formats can be defined to suit local clinical practice, to accommodate particular combinations of monitoring and measuring equipment, and to support local innovation (Morrison, Blackwell & Vuylsteke 2010). As a result of this substantial degree of customisability, the data extraction tool provided with the product (called 'Query Wizard') needs to be highly generic, allowing for queries that match and extract any possible combination of data types and values. This extremely high degree of abstraction and customisability results in a cognitive dimensions profile closer to that of a professional programmer than typical end-user tools.

Documentation is limited, and those key users who do use the tools are self-taught, because formal training is not readily available. The need for data extraction has steadily increased in recent years so that those key users who have developed skills over time have offered to teach others in sites across the national health care provider. So cumbersome is the clinical information system data extraction process that when investigating small data sets e.g. a single or small group of patients, users typically choose to, or are often encouraged to, read data off the screen of the user interface intended for everday observation and data entry, rather than attempt an automated data extraction process.

Aside from the difficulty in using extraction tools, an additional concern is that any data extracted using the tool is in a structure that does not lend itself to immediate analysis. Data is distributed in such a way that it is necessary for key users and/or end users to first manipulate the data set so that it is arranged in rows and columns that can be used to address the initial data request. In order to do this users turn to an intermediate tool i.e. spreadsheet software, to first organise data into suitable views. This is particularly necessary when dealing with large, more complex data requests. Given the limitations of the extraction tools, such queries need to be sub-divided into queries key users estimate will be executable. Each of these sub-queries is then aggregated and manipulated in a spreadsheet to build up the overall data query result.

Over time, key users have developed other ways to overcome data extraction issues. For example, one key user has kept a separate data set in a spreadsheet (manually updated at intervals) which enables her to more quickly query basic data in a format that requires less, and often no, manipulation after the query process.

Another way users have tried to overcome these issues is to circumvent the use of the CIS data extraction tools altogether. A subset of data collected in the CIS has to be submitted to the national

intensive care audit database ICNARC. This data is either extracted from the CIS automatically, or (sometimes) partially manually re-keyed into the required audit data set. Despite the challenges that this extraction process itself introduces, the ICNARC data set has a well-defined structure (as required by national audit specifications) which lends itself to being more easily queried than the CIS. Both key users and end users will use this data set in their local contexts to perform queries which they regard as a simpler process by comparison to using the CIS. Evidently the key constraint is that this is a subset of the entire CIS database and so can only support particular queries.

As alluded to previously, a constraint for key users is the limited level of skills they may have to perform data related tasks. While it is rare for key users to have data-related training, a few have made efforts to develop specific programming skills, such as SQL, to extend their ability to extract and manipulate data in more useful ways. In effect this enables them to bypass the CIS and the related data extraction tools, allowing them to directly access the database. Such efforts are still limited, however, because key users have a relatively basic understanding of the CIS database structure, thus placing a constraint on the effectiveness of data extraction.

A related issue limiting the ability of key users to service data requests is that they must typically combine such tasks with a wider set of responsibilities. Work on CIS tasks has to be balanced with, for example, clinical or other IT related work. This limits the amount of time key users can dedicate to working with the CIS as they prioritise more urgent work accordingly. Key users often have to make expeditious decisions as to whether it is possible to service a data request. If a request appears too complex and time consuming to extract and manipulate the data, it is likely to be deferred or rejected when the wider responsibilities of the key user are considered more urgent.

We observe that the consequence of this prioritisation is that end users may limit data requests over time. Knowing that key users are busy, or that a data request is likely to be too complex, results in end users either not asking the question at all or perhaps agreeing to simplify the question to match the time and abilities of the key user as well as the capabilities of the CIS, pre-empting the negotiation they are likely to have with the key user in agreeing the data request.

3.2. Data semantic roles

As mentioned in the previous section, a distinctive (and desirable) feature of this CIS is that the recorded data is fully customisable (Morrison, Blackwell & Vuylsteke 2010). This is achieved through use of an extremely general database model. The main data store is a single table of time-stamped entries, each consisting of an attribute-value pair. The set of possible attribute values is completely configurable in each installation of the system, defined as the set of ‘variables’ that can be recorded, reported, graphed or acted upon.

Each entry in a table represents a single observation – the type of observation or measurement that has been made, and the observed result or value of the measurement. This logic is compatible with the logic of the critical care unit, in which the clinical team are constantly making observations or measurements of the patients. However the practical application of these observations is constrained by a number of practical issues that arise from the characteristic ways data is collected for analysis. These different natural categories of data observation can be compared to the ‘roles of variables’ identified in conventional code (Sajaniemi & Prieto 2005), for example:

1. Statistical invariants: Apart from the changing observations, most other entries in the database are single values, not expected to change. We are not asserting that these things can never change, simply that they are invariant for the purposes of clinical reasoning. Examples include data of birth, height, eye colour etc.

2. Timebase resampling: Various observations are collected at differing intervals. One aspect of a patient's condition might be recorded once a day, another aspect three times a day, another every hour, and another (automatically measured) every five minutes. Whenever it is necessary to compare or relate these observations, some kind of interpolation is necessary, whether as simple as using two values as close to each other as possible (constant nearest neighbour), linearly interpolating between points on either side, or fitting a regression model to account for sources of variance.

3. Persistent legacy data: When new variable types are added to the local CIS configuration, it is not possible to delete the previous variables (because this would invalidate existing queries and reports). Data associated with those variables is therefore retained in the database, with local practices used as work-arounds to prevent confusion (for example, renaming out-of-date attributes with the letter 'z' as a prefix, so that they are not inadvertently selected when attribute values are offered in alphabetical order).

3.3. Data ontology

Our focus in this project on a specific domain has highlighted the ways in which the degree of abstract generalisation supported by customisable tools might be unhelpful. This builds on observations with regard to design-time abstraction (Blackwell et al 2008), and also on the suggestion that external representations can be more usable if made less abstract, so that users are able to reason about more specific interpretations (Stenning and Oberlander 1995). These considerations are aspects of the cognitive dimension of abstraction, which we believe is a central issue in the respective approaches of key users and data end users. In this section, we therefore offer some further analysis of the types of abstraction that have resulted from the highly customisable CIS system.

In the relational model, data types are ontologically undifferentiated, with attention paid only to the compatibility of machine data types, the specification of whether a value might be used as a relational key, and possibly special viewing or editing facilities for free text or binary objects. We speculate that in an end-user context such as the repurposing of patient data, this extreme degree of abstract generalisation is unhelpful. For example, the name of a patient "Adam Smith" and the name of a surgical procedure "coronary artery bypass graft" are both text fields. However, there is no clinical context in which it would be meaningful to compare these two values to each other, or to count the relative numbers of each. Our hypothesis is that manipulation of data will be easier for end-users to achieve if values such as these cannot be confused, by means of detecting and maintaining the ontological referents of different types of data in the database.

The design of programming standards to reflect ontological categories is reminiscent of the People, Places and Things design standard promoted by Apple's Taligent spin-out in the 1990s (Cotter & Potel 1995). Taligent failed in part because of failure to anticipate that most software 'objects' would not be 'things', but rather purely engineering abstractions such as lists, buffers, wrappers, sockets and so on (Blackwell 1993). More seriously, perhaps, they were unable to enforce a business model in which a single understanding of ontology could be imposed on potentially incommensurable knowledge systems (Star & Bowker 1999). Attempts to create ontologically-justified data type protocols continued in initiatives such as the Object Management Group's Business Object Model (BOMSIG) (Zamir 1998), or continuing Semantic Web work of the W3C Web Ontology Language (McGuinness & Van Harmelen 2004). Arguably, all of these share the fallacy identified by Umberto Eco as the "Search for the Perfect Language" (1995).

A lighter-weight approach to ontology and types can be found in Scaffidi's Topes proposal (Scaffidi et al 2008), as well as in the unit inference approach to spreadsheets developed by Abraham and Erwig (2004). Ontological differentiation might recognise common sense categorisations, such as TIME IS ORDERED. CAUSALITY IS UNIDIRECTIONAL, and PERSONS ARE INDIVIDUALS. These categories also aid statistical reasoning. For example, it is reasonable to investigate whether the time of day for surgery has any consistent effect on the outcome of an operation. It is less reasonable to investigate whether the time of day for surgery has a consistent effect on the patient's date of birth. By taking the ontological referents of data into account, the tabular structure of the data can help to guide statistical enquiries.

4. Potential design strategies

We propose a design for a system, which guides the user through constructing a simple example explaining their intent, automatically synthesises a likely spreadsheet transformation program which matches this example, and then executes the program, having the rest of the spreadsheet as an input. Below, we present our approach by describing its three dimensions, following best practises of the program synthesis community (Gulwani, 2010): the way the user shows what their intent is, the space of possible programs, and strategy for searching the space of possible programs.

4.1. Specifying the intended program

One problem with many menu-driven interfaces for data wrangling is the necessity for the user to have clear idea of the meaning of each of the suggested transformation steps. Naturally, this means a steep learning curve, making it hard for novices to use the system. In the context we have been studying, it is apparent that CIS users such as nurses and doctors often lack the time and access to a technical person who might help them learn how to use new software. Rather than menu-driven specification, we have therefore focused on the development of a demonstration-based approach, in which users interact directly with the data to select the items that are of interest. The interaction method that we have created, responding to this requirement, is called Data Noodles, and is described in a companion paper (Gorinova, Sarkar and Blackwell 2016)

4.2. Transformation language

The transformation of spreadsheets, selecting and restructuring them in response to the user's actions, is achieved with a Domain Specific Language (DSL) largely inspired by the Potter's Wheel and Wrangler's transformation languages. It implements reshaping transformations, such as Fold and Unfold, substitution of missing values, and we plan to add support for data value interpolation, filtering, splitting and merging operations. We choose to use a language adapted from the two above, as this previous work has found them sufficiently expressive for common transformations, and furthermore, providing greater readability than more 'fine-grained' languages in which the program is a collection of constraints determining the value of each cell in the output spreadsheet (Barowy et al., 2015; Harris & Gulwani, 2011).

A key requirement for the CIS context is an *interpolation* operation, which fills in missing values of a column, by estimating them based on some function. In the context of electronic health records this is particularly useful, as often timestamps of different continues parameters do not match, e.g. heart rate could be recorded every minute, but respiratory rate every five minutes. Thus, if a clinician wants to analyse a dependency between a patient's heart and respiratory rates, they either need to delete all extra heart rate entries (which will work only if the two time-series have some data points taken at the same time) or they could speculate about the way parameters behave between measures. Our interpolation operator would offer a range of options for describing this speculated behaviour, such as 'copy the last known value', 'copy the nearest known value', 'linearly interpolate', etc.

4.3. Program synthesis

To synthesise programs in the DSL described above, we have been using the PROSE SDK (Polozov & Gulwani, 2015). PROSE is a .NET framework which provides tools for defining the syntax and semantics of a domain-specific language and can then synthesise a ranked set of programs satisfying some input-output example specification. To be able to synthesise spreadsheet transformation programs with PROSE, we define the following:

- Syntax: a text file defining the syntax of the transformation languages as a context free grammar.

- Semantics: a static C# class defining the *operational semantics* of each symbol of the grammar. That is an actual implementation of what each of the spreadsheet transformation language operations does.

- Witness Functions: a static C# class defining the *inverse semantics* of each symbol in the grammar. A witness function deduces information for a parameter of a function, given some information about the entire function. For example, consider the column renaming function `Rename(S, OldColumnName, NewColumnName)`, which returns `S'` – the spreadsheet `S`, but with the name of the column `OldColumnName` changed to `NewColumnName`. A witness function of `Rename` for `S` is then:

$$W_S(S') = \texttt{Rename(S', NewColumnName, OldColumnName)}$$

- Scoring Functions: a static C# class defining a *scoring function* of each symbol in the grammar. That is, we inductively define a score of the entire transformation program, based on its components. For example, we adopt an Occam's razor approach -- the longer a program is, the lower its score is, as we believe that the most probable program the user wanted, is the simplest one that explains the input-output example.

The above definitions allow PROSE to find spreadsheet transformation programs that are likely to describe the user's intent, based on the example the user has provided.

5. Discussion

We have presented a context in which users face a very specific end-user programming problem arising from the need to restructure data from one loosely constrained context (a configurable information system) to another (exploratory statistical analysis). It appears that many of the challenges they face could be addressed by an example-based interaction approach, in which they would demonstrate the data format that they need, and that this demonstration could be used to synthesise a data transformation program. We have presented an overview of a program synthesis approach that could be used for this purpose, if made sufficiently accessible.

In a companion paper (Gorinova et al 2016), we propose an interaction paradigm called Data Noodles, that we hope will be accessible to users, while offering sufficient power to achieve the data transformations appropriate to this application domain. In future, we plan to integrate ontological representation elements into this paradigm, in a way that will help constrain the statistical analyses that might then be carried out with such data.

6. Acknowledgements

This research is funded by the Health Foundation.

7. References

Abraham, R., & Erwig, M. (2004). Header and unit inference for spreadsheets through spatial analyses. In Proc VL/HCC 2004, pp. 165-172).

Barowy, D. W., Gulwani, S., Hart, T., & Zorn, B. (2015). FlashRelate: Extracting relational data from semi-structured spreadsheets using examples. ACM SIGPLAN Notices, 50(6), 218–228.

Blackwell, A.F. (1993). Bottom-Up Design and this Thing called an 'Object' .EXE Magazine, 8(7), 28-32.

Blackwell, A.F. (2001). See What You Need: Helping end users to build abstractions. Journal of Visual Languages and Computing, 12(5), 475-499.

Blackwell, A.F. and Morrison, C. (2010). A logical mind, not a programming mind: Psychology of a professional end-user. In Proc PPIG 2010, pp. 175-184.

Blackwell, A.F., Church, L. and Green, T.R.G. (2008). The abstract is 'an enemy': Alternative perspectives to computational thinking. In Proceedings PPIG'08, pp. 34-43.

Church, L. and Blackwell, A.F. (2008). Structured text modification using guided inference. In Proceedings PPIG'08, pp. 83-94.

Cotter, S., & Potel, M. (1995). Inside Taligent technology. Addison-Wesley.

Eco, U. (1995). The search for the perfect language. Wiley-Blackwell.

Fisher, K., & Walker, D. (2011). The PADS project: An overview. In Proceedings of the 14th International Conference on Database Theory - ICDT '11 (p. 11). ACM Press.

Galhardas, H., Florescu, D., Shasha, D., & Simon, E. (2000). AJAX: an extensible data cleaning tool. ACM SIGMOD Record, 29(2), 590.

Gorinova, M., Sarkar, A. and Blackwell, A.F. (2016). Transforming Spreadsheets with Data Noodles. Submitted to VL/HCC 2016.

Gulwani, S. (2010). Dimensions in program synthesis. In Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming - PPDP '10 (pp. 13–24). ACM Press.

Guo, P. J., Kandel, S., Hellerstein, J. M., & Heer, J. (2011). Proactive wrangling: Mixed-initiative end-user programming of data transformation scripts. In Proc UIST '11, p. 65

Harris, W. R., & Gulwani, S. (2011). Spreadsheet table transformations from examples. In Proc ACM PLDI '11 (Vol. 46, p. 317)

Kandel, S., Paepcke, A., Hellerstein, J., & Heer, J. (2011). Wrangler: Interactive visual specification of data transformation scripts. In Proc CHI 2011, pp. 3363-3372.

Le, V., & Gulwani, S. (2014). FlashExtract: A framework for data extraction by examples. ACM SIGPLAN Notices, 49(6), 542–553.

Lin, J., Wong, J., Nichols, J., Cypher, A., & Lau, T. A. (2008). End-user programming of mashups with Vegemite. In Proceedingsc of the 13th international conference on Intelligent user interfaces - IUI '09 (p. 97). ACM Press.

McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. W3C recommendation, 10(10), 2004.

Morrison, C & Blackwell, A.F. (2009) Observing end-user customisation of electronic patient records. In Proc. 2nd International Symposium on End-User Development, IS-EUD'09, pp. 275-284.

Morrison, C., Blackwell, A.F. and Vuylsteke, A. (2010). Practitioner-customizable clinical information systems: a case-study to ground further research and development opportunities. *Journal of Healthcare Engineering* **1**(3), 297-314.

Polozov, O., & Gulwani, S. (2015). FlashMeta: a framework for inductive program synthesis. In Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA 2015 (Vol. 50, pp. 107–126). ACM Press.

Raman, V., & Hellerstein, J. M. (2001). Potter's wheel: An interactive data cleaning system. In VLDB (Vol. 1, pp. 381–390).

Sajaniemi, J., & Prieto, R. N. (2005). Roles of variables in experts' programming knowledge. In *Proceedings (PPIG 2005,* pp. 145-159.

Sarkar, A., Blackwell, A., Jamnik, M., & Spott, M. (2014, July). Teach and try: A simple interaction technique for exploratory data modelling by end users. In Proc VL/HCC 2014, pp. 53-56

Scaffidi, C., Myers, B., & Shaw, M. (2008). Topes: reusable abstractions for validating data. In Proceedings of the 30th int ACM conf on Software engineering, pp. 1-10).

Star, S. L., & Bowker, G. (1999). Sorting Things Out: Classification and its Consequences.

Stenning, K., & Oberlander, J. (1995). A cognitive theory of graphical and linguistic reasoning: Logic and implementation. Cognitive science, 19(1), 97-140.

Tuchinda, R., Szekely, P., & Knoblock, C. A. (2008). Building Mashups by example. In Proceedings of the 13th international conference on Intelligent user interfaces - IUI '08 (p. 139)

Verborgh, R., & Wilde, M. De. (2013). Using OpenRefine. Packt Publishing Ltd.

Zamir, S. (Ed.). (1998). Handbook of object technology. CRC Press, p. 49-3