Towards a Spreadsheet-Based Language Workbench

Mikhail Barash
Bergen Language Design Laboratory, *University of Bergen*, N-5020, Bergen, Norway
mikhail.barash@uib.no

***Abstract*—Spreadsheets are widely used across industries for various purposes, including for storing and manipulating data in a structured form. Such structured forms—expressed using tabular notation—have found their way in language workbenches, which are tools to define (domain-specific modeling) languages and Integrated Development Environments (IDE) for them. There, a tabular notation is oftentimes used as a secondary way to represent concrete syntax of certain language constructs; however, it is not a primary means for (meta)model definition. We present early results on implementing a language workbench where metamodels, models, and editor services are defined only using a tabular notation. We give an overview of the desired functionality of spreadsheet-based language workbenches.**

***Index Terms*—Spreadsheets, Microsoft Excel, language workbench, tool support, domain-specific modeling.**

I. Introduction

One of the use cases for a spreadsheet is to structure the data—using a *tabular notation*—and then to store and manipulate it in that structured form. Addressing this are approaches that support model-driven development within spreadsheets (e.g., [3]–[5], [9], [10]). More traditional textual and graphical modeling languages are oftentimes implemented using *language workbenches* [7], which are tools to define—using one ore more metalanguages—languages' syntax, semantics, and tool behaviour. A language workbench outputs a tailored IDE with *editor services*, such as syntax-aware editing, syntax highlighting, code formatting, completion, navigation, corrections [7]. This facilitates adoption of a language [7].

Although both spreadsheets and language workbenches can be used for metamodeling, there is a gap between them stemming from their different primary focus: namely, tabular notation and tool support. Indeed, in language workbenches, can all the (meta)languages only use tabular notation? And in the context of spreadsheet-based metamodeling, can the behaviour of a spreadsheet tool be specified first-class? The goal of this paper is to address this two-fold gap. We introduce a *spreadsheet-based language workbench MetaTabular*, currently being implemented as an add-in for Microsoft Excel.

II. Excel as a Language Workbench

The language definition in MetaTabular is inspired by that of MPS [2]. A language is a set of *concepts* (language constructs), and for each concept, one can define its abstract syntax, concrete syntax, validation logic, and tool behaviour, using corresponding metalanguages. In MetaTabular, these metalanguages only use a tabular notation. Definition of abstract syntax—*structure*—includes the name of a concept and a list of its *members*: values owned by the concept and aggregation relationships with other concepts, as shown in Fig. 1(a,b). A concept can inherit another concept's members (see cells A2 and B15 in Fig. 1(b)). Definition of a concrete syntax—*presentation*—is a template for cell layout of concept instances, as shown in Fig. 1(a,b). Templates can specify cells that have fixed values (e.g., cell I10 in Fig. 1(b)), refer to concept's members (J10), etc. Each presentation has a *handle* (I9, I15)—a textual representation of a concept instance, which appears, e.g., in autocomplete menus (see cell D18 in Fig. 1(c)). A presentation can also specify *editor actions* for concept instances. Fig. 1(a) shows a definition of an action that converts member `name` of concept `Person` to upper case. The action definition includes its name (H11), target member (I11), a formula executing the action ("`=TOUPPER(I11)`" in J11), and a description (K11). Cells K6:M6 specify a button that will appear at model-time (cf. cells F6:H6 in Fig. 1(c)). One can also define custom command panes with macros written in Excel's JavaScript, as shown in Fig. 1(f).

Functionality	Motivation
the workbench is implemented within a spreadsheet rather than as a stand-alone tool	shallow learning curve; users benefit from spreadsheet's rich formatting and visualization functionality
metamodels, models, editor services are defined explicitly in a tabular notation	familiar notation; "reuse by copy-paste"; no implicit definitions
editor services defined in a tabular notation	users can extensively customize the spreadsheet tool behaviour
metamodels, models, editor services definitions are synchronized	changed definitions are automatically "propagated" and re-validated
modular definitions using a tabular notation	enabling language composition [6], language migration [2]
model transformations specified in a tabular notation	enabling code generation; refactoring [8]
interoperability with other language workbenches	import, export; also: editing Excel-based models in EMF tools [11]
bootstrapped workbench	users can create their own metalanguages and modify existing ones

TABLE I
Desired functionality of spreadsheet-based workbenches.

Validation rules are specified in concept's structure and presentation. In Fig. 1(b), cell E10 specifies that member `id` of concept `Book` must conform to a regular expression. Cell E13 enlists measurement units for member `price`. Cell E12 specifies that `author` is a reference to an instance of concept `Author`. A metamodeler can define custom validation rules, as shown in Fig. 1(a): cell J6 specifies an error message that appears when `name` of `Author` is an empty string.

MetaTabular provides a contextual support for language definitions, e.g., outline views for (meta)models, as shown

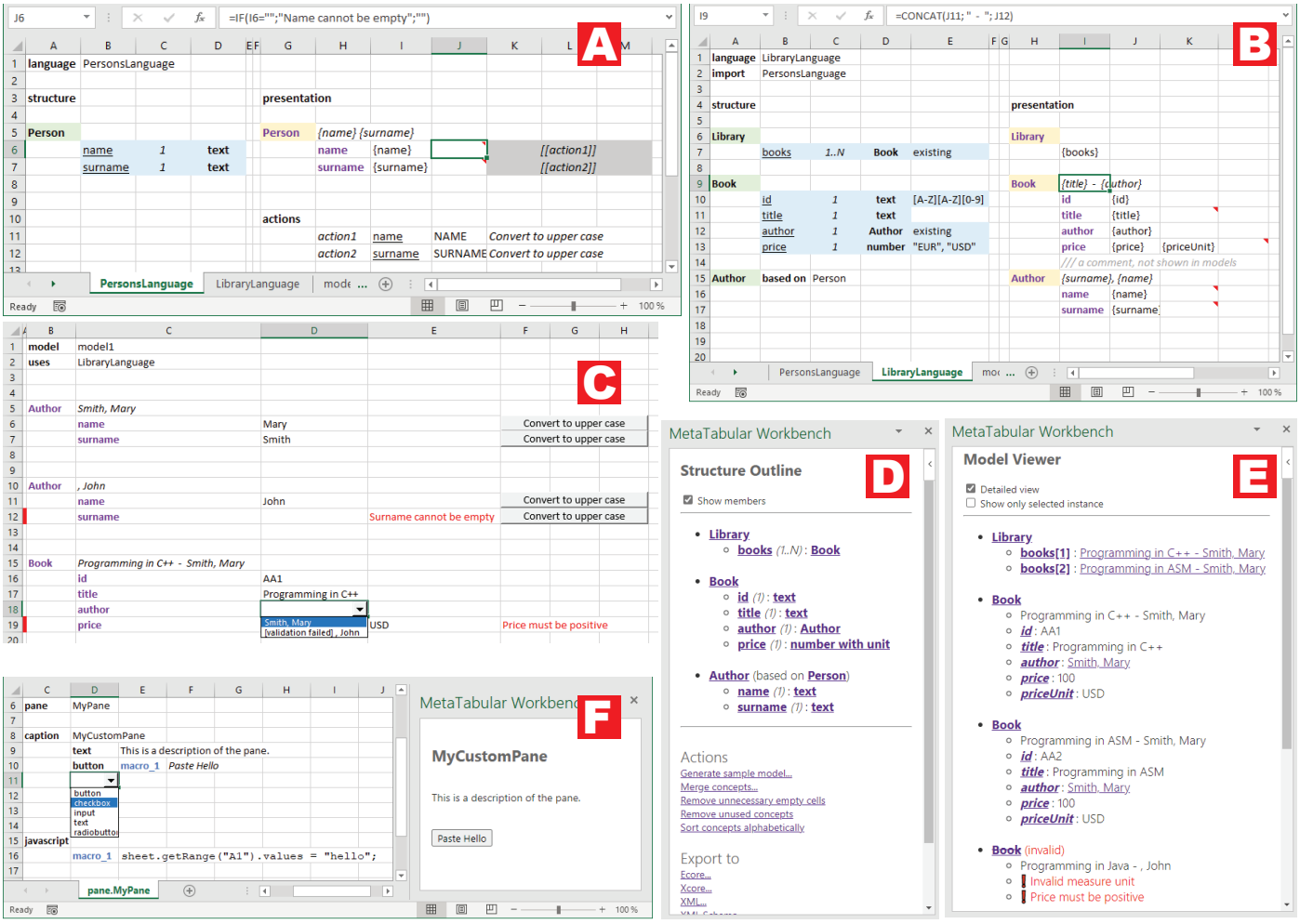

Fig. 1. *(a,b)* metamodel definitions; *(c)* a model definition; *(d,e)* (meta)model navigation via outline view; *(f)* a user-defined command pane.

in Fig. 1(d,e). (Meta)models could be imported from and exported to a number of formats (e.g., Ecore, XML, JSON).

III. Related Work

There is an extensive body of literature on enabling modeling within spreadsheets: metamodels can be defined both outside [5] and within [4] a spreadsheet software, or are inferred from spreadsheet data [3], [10]. These approaches are not however centered around editor services, which in MetaTabular are defined first-class using a tabular notation.

Currently, MetaTabular follows a naïve parsing strategy for (meta)languages' cell layout; we may benefit from results on parsing domain-specific notations [1] and a formalization of template layouts [5].

IV. Discussion

MetaTabular is an initial step in understanding the limitations of bare tabular notations for defining (meta)models *and* editor services; we see an apparent connection with projectional language workbenches [2], [7], where concrete syntax of (meta)languages is defined as a collection of cells, although with a traditional text editor's editing experience.

Of the desired functionality of spreadsheet-based language workbenches given in Table I, the most important is perhaps *bootstrapping* [7]. Its successful implementation will substantially facilitate creation of spreadsheet-based workbenches.

References

[1] S. Adam, U. P. Schultz, *Towards tool support for spreadsheet-based domain-specific languages*. GPCE 2015. 95–98.

[2] F. Campagne, *The MPS Language Workbench, Vol. 1*. 2014.

[3] J. Cunha, M. Erwig, J. Mendes, J. Saraiva, *Model inference for spreadsheets*. Autom. Softw. Eng. 23(3). 361–392. 2016.

[4] J. Cunha, J. P. Fernandes, J. Mendes, R. Pereira, J. Saraiva, *MDSheet – Model-Driven Spreadsheets*. CEUR Workshop Proceedings 1209. 2014.

[5] G. Engels, M. Erwig, *ClassSheets: automatic generation of spreadsheet applications from object-oriented specifications*. ASE 2005. 124–133.

[6] S. Erdweg et al., *Language composition untangled*. LDTA 2012. 7.

[7] S. Erdweg et al., *Evaluating and comparing language workbenches: Existing results and benchmarks for the future*. Comput. Lang. Syst. Struct. 44. 24–47. 2015.

[8] F. Hermans, D. Dig, *BumbleBee: a refactoring environment for spreadsheet formulas*. SIGSOFT FSE 2014. 747–750.

[9] F. Hermans, R. F. Paige, P. Sestoft (Eds.), *Software Engineering Methods in Spreadsheets*. CEUR Workshop Proceedings 1209. 2014.

[10] F. Hermans, M. Pinzger, A. van Deursen, *Automatically Extracting Class Diagrams from Spreadsheets*. ECOOP 2010. 52–75.

[11] I. Ráth, *EMFxcel: Having Fun with Excel and Eclipse Modeling Tools*. Available at: http://viatra.inf.mit.bme.hu/incquery/examples/emfxcel.