SAPLING 2011 Abstract
Gary Miller
CTO Sumwise.com

Closing trapdoors, taking the meta out of the menu, and dynamic nodes.

The spreadsheet paradigm[1] introduced by VisiCalc is over 30 years old. Spreadsheets are arguably the world's most popular programming paradigm and its rate of growth is outstripping any professional programming language[2]. However, spreadsheets have hardly changed since their inception, they are not Turing-complete[3], they are horribly error prone[4], and *"lack the most fundamental mechanism that we use to control complexity: the ability to define re-usable abstractions"*[5].

This presentation is about work done to strengthen spreadsheet modelling and proposals for further work. The two guiding forces for this work have been: 1) encapsulating the design patterns used by expert spreadsheet modellers; and 2) learning from existing programming languages and adding to the spreadsheet paradigm where appropriate whilst being cognisant of the human factors.[6]

Taking the meta out of the menu. References by address, and how this can be used declaratively to achieve goal seek and Turing-completeness.

Closing trapdoors. The trapdoor as defined by Peyton Jones et al., is the dropping into another language / paradigm for the user to define functions. The primary feature we have added is the ability to group cells together via a declarative group definition. In order for these definitions to be powerful enough we needed to add a number of features as first-class citizens (named nodes, structured axes and label meta-data on nodes). Cell groups are a strong design pattern in spreadsheet modelling as can be observed by the behavior of expert modellers. Functions are defined by simple annotation of some groups as arguments or returns.

Dynamic Nodes — an extension to named nodes (rows, columns and higher dimension axis nodes) — allow a node to contain a formulae (or reference) and therefore the represent a dynamic number of nodes. It is possible that a formula could be recursive and add another avenue to Turing-completeness (translation to Lambda calculus).[7] In the absence of an implementation of Dynamic Nodes the appropriateness of this feature for End-user programmers in unknown, and it may turn out to only be useful for library writers.

---

[1]Also refered to as: - Programming by Demonstration, Interactive programming http://www.cs.bham.ac.uk/~rnp/, Example Centric Programming http://subtextual.org/OOPSLA04.pdf, Illustrative Programming http://martinfowler.com/bliki/IllustrativeProgramming.html

[2]Scaffidi, C., Shaw, M., and Myers, B. (2005) Estimating the numbers of end users and end user programmers. IEEE Symposium on Visual Languages and Human-Centric Computing, 207-214.

[3]Assuming the exclusion of circular references, and Macro languages (the trapdoor)

[4] Panko, "What We Know About Spreadsheet Errors" http://panko.shidler.hawaii.edu/ssr/Mypapers/whatknow.htm

[5]A User-Centred Approach to Functions in Excel, Peyton Jones et al.

[6]Formally Cognitive Dimensions framework and the Attention Investment model. Informally thinking about the what spreadsheets have gotten right in terms of human cognition (eg. 2D, recognitions of names)

[7]This may be equilant to "First Class Copy & Paste" Jonathan Edward http://dspace.mit.edu/bitstream/handle/1721.1/32980/MIT-CSAIL-TR-2006-037.pdf?sequence=1