2007 IEEE Symposium on Visual Languages and Human-Centric Computing, USA, Sept 23-27 2007, © IEEE 2007

MaramaTatau: Extending a Domain Specific Visual Language Meta Tool with a Declarative Constraint Mechanism

Na (Karen) Liu[1], John Grundy[1, 2] and John Hosking[1]
Department of Computer Science[1] and Department of Electrical and Computer Engineering[2], University of Auckland, Private Bag 92019, Auckland, New Zealand
{karen | john-g | john }@cs.auckland.ac.nz

Abstract

It is increasingly common to use meta tools to specify and generate domain specific visual language tools. A common problem for such meta tools is the specification of model level behaviours, such as constraints and dependencies. These often need to be specified using conventional code in the form of event handlers or the like. We report our experience in integrating a declarative constraint/dependency specification mechanism into a domain specific visual language meta tool, focussing on the tradeoffs we have made in the notational design and environmental support used. The expressive power of the mechanism developed is illustrated by a substantial case study where we have redeveloped a complex visual tool for architectural modelling, eliminating conventional event handlers.

Keywords: visual constraint language, visual language meta tool

1. Introduction

Brief on need for behavioural spec in meta tools, ways of doing so

2. Background and Motivation

Elaborate on the above. Lit review

Introduce Marama/Pounamu. Talk briefly about Kaitiaki and approach to solve constraint/behaviour at UI level – need for model level equivalent. Examples of existing code in MaramaMTE handlers or similar.

Come up with set of requirements focussing on how to best incorporate a declarative constraint/dependency mechanism into a domain specific visual language metatool that allows:

simple specification consistent with but complementary to the existing visual metaphor

simple debugging/visualisation consistent with the metaphor

avoids the need to write java code/event handlers for model level constraints/behaviour

3. MaramaTatau (our approach)

Introduce notation using whole part example

Talk about design decisions made

- sacrificed concreteness (split specification and implementation more than say Forms 3) due to need to support complex abstractions
- nevertheless a fairly straightforward relationship with spreadsheet metaphor in terms of runtime behaviour

4. Case study

MaramaMTE example

5. Implementation

Details of implementation – use of OCL package. Details of architecture

6. Discussion and Evaluation

Elaborate more on tradeoffs/design decision tying back to Cog Dimensions

Elaborate on case study as a demonstrate by example evaluation. Talk about amount of code reduction, increased code visibility, etc

Indicate need to do end user evaluation

7. Conclusions and future work

Mention fits in with broader generalisation of event management – to come

Acknowledgements

FRST

References

[1]