
Proceedings Paper:
Paige, Richard F. orcid.org/0000-0002-1978-9852 (2016) Language engineering:Challenges, opportunities and potential disasters for interactive systems. In: EICS 2016 - 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2016, 21-24 Jun 2016 ACM, BEL, p. 3.

https://doi.org/10.1145/2933242.2948132

Language Engineering: Challenges, Opportunities and Potential Disasters for Interactive Systems

Richard F. Paige
University of York
York, United Kingdom
richard.paige@york.ac.uk

ABSTRACT
Language engineering underpins model-driven engineering and the application of domain-specific languages. In this talk, I will introduce language engineering and its principles and practices, using model-driven engineering as an exemplar. I will suggest how the engineering of interactive systems offers opportunities, challenges and the potential for chaos for language engineering.

Author Keywords
Language engineering; Model-Driven Engineering; domain-specific languages

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Language engineering involves building artificial (software) languages to solve problems. Very often, the languages we build are applied to solve problems of *automation*, e.g., to support automating repetitive and error-prone tasks efficiently and effectively. A wealth of technologies have been developed to support language engineering, ranging from grammars and parsing algorithms, to metamodeling infrastructure through to model transformation and code generation tools.

Software languages are themselves interactive systems: they generally come with editors (e.g., GUIs or IDEs) that provide useful features, including syntax checking and highlighting, code completion, refactoring etc.

This talk explores the issue of how we can build better software languages – focusing on interactions with their users. I will talk about a typical traditional software language engineering process, based on Model-Driven Engineering technology, and will draw out some of the challenges associated with ensuring that we can think about interacting with languages (and their supporting tools) as the languages are being constructed. I will illustrate this approach with an example of a real system developed to support air traffic controllers. I will then present an alternative approach to language engineering that may offer more flexibility in terms of evaluating the effectiveness of user interactions with their supporting tools. I will attempt to synthesise some of the key open challenges associated with language engineering in practice, with some focus on issues related to their application to engineering interactive systems.

REFERENCES
1. Richard F. Paige, Dimitrios S. Kolovos, Fiona A. C. Polack: A tutorial on metamodelling for grammar researchers. *Sci. Comput. Program.* 96: 396-416 (2014)
2. Dina Salah, Richard F. Paige, Paul A. Cairns: Observations on Utilising Usability Maturity Model-Human Centredness Scale in Integrating Agile Development Processes and User Centred Design. *SPICE 2015*: 159-173
3. Athanasios Zolotas, Nicholas Drivalos Matragkas, Sam Devlin, Dimitrios S. Kolovos, Richard F. Paige: Type Inference in Flexible Model-Driven Engineering. *ECMFA 2015*: 75-91
4. Steffen Vaupel, Daniel Strüber, Felix Rieger, Gabriele Taentzer: Agile Bottom-Up Development of Domain-Specific IDEs for Model-Driven Development. *FlexMDE@MoDELS 2015*: 12-21.