Natural Language Engineering **10** (3/4): 205–209.
DOI: 10.1017/S1351324904003481 Printed in the United Kingdom

Software Architecture for Language Engineering

HAMISH CUNNINGHAM

Department of Computer Science, University of Sheffield, Sheffield, S1 4DP, UK
e-mail: hamish@dcs.shef.ac.uk

DONIA SCOTT

ITRI, University of Brighton, Lewes Road, Brighton, BN2 4GJ, UK
e-mail: Donia.Scott@itri.brighton.ac.uk

(*Received 1st June 2004*)

1 Software Architecture

Every building, and every computer program, has an *architecture*: structural and organisational principles that underpin its design and construction. The garden shed once built by one of the authors had an ad hoc architecture, extracted (somewhat painfully) from the imagination during a slow and non-deterministic process that, luckily, resulted in a structure which keeps the rain on the outside and the mower on the inside (at least for the time being). As well as being *ad hoc* (i.e. not informed by analysis of similar practice or relevant science or engineering) this architecture is *implicit*: no explicit design was made, and no records or documentation kept of the construction process.

The pyramid in the courtyard of the Louvre, by contrast, was constructed in a process involving *explicit design* performed by qualified engineers with a wealth of theoretical and practical knowledge of the properties of materials, the relative merits and strengths of different construction techniques, *et cetera*.

So it is with software: sometimes it is thrown together by enthusiastic amateurs; sometimes it is *architected*, built to last, and intended to be 'not something you finish, but something you start' (to paraphrase Brand (1994)).

A number of researchers argued in the early and middle 1990s that the field of computational infrastructure or architecture for human language computation merited an increase in attention. The reasoning was that the increasingly large-scale and technologically significant nature of language processing science was placing increasing burdens of an engineering nature on research and development workers seeking robust and practical methods (as was the increasingly collaborative nature of research in this field, which puts a large premium on software integration and interoperation). Over the intervening period a number of significant systems and practices have been developed in what we may call Software Architecture for Language Engineering (SALE).

This special issue represented an opportunity for practitioners in this area to report their work in a coordinated setting, and to present a snapshot of the state-of-the-art in infrastructural work, which may indicate where further development and further take-up of these systems can be of benefit.

2 The Papers

The SALE systems reported in this issue can be categorised according to a number of dimensions, such as the broad sub-field(s) of language computation supported, the component model adopted, the support for testing and evaluation provided, or the significance given to shared data structures. One thing that they all have in common is that they produce and consume **language resources**. Ide and Romary report the creation of a framework for linguistic annotations as part of the work of ISO standardisation Technical Committee 37, Sub-Committee 4, whose objective

> . . . is to prepare various standards by specifying principles and methods for creating, coding, processing and managing language resources, such as written corpora, lexical corpora, speech corpora, dictionary compiling and classification schemes. These standards will also cover the information produced by natural language processing components in these various domains.[1]

The work reported here is from Working Group 1 of the committee, which has developed a linguistic annotation framework based on the XML, RDF(S), and OWL.

The literature on SALE has historically been weighted towards language analysis as opposed to generation; to redress the balance a little we have two papers arising from work on the Reference Archicture for Generation Systems (RAGS). Mellish *et al.* present the RAGS conceptual framework, while Mellish and Evans discuss the implementation of this framework in several experimental systems, and how these systems illustrate a range of wider issues for the construction of SALE for generation.

One of the most demanding areas for infrastructural work in recent years has been that of dialogue management systems. Herzog *et al.* present the latest in three generations of architecture to arise from the Verbmobil and Smartkom projects, in the shape of the Multiplatform system. This architecture supports multiple distributed components from diverse platforms and implementation languages running asynchronously and communicating via a message passing substrate.

A persistent theme in this journal has been measurement, quantitative evaluation and the relationship between engineering practice and scientific theory. To quote Kelvin:

> When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the stage of science.
> (Lord Kelvin (William Thomson), in a lecture to the Institution of Civil Engineers, London, 3 May 1883.)

On the other hand, Einstein tells us:

> Not everything that counts can be counted, and not everything that can be counted counts.
> (Albert Einstein, from a sign hanging in his office at Princeton University.)

[1] `http://www.tc37sc4.org/`

SALE work has taken similarly varied approaches to measurement, both of component systems developed using SALE systems and of the success of those systems themselves. The presentation of IBM's TEXTRACT architecture (Neff *et al.*) includes an illustration of how the same mechanism can be used for producing both quantitative metrics and for visual feedback to users of the results of automated processing.

Ferrucci and Lally report a successor to TEXTRACT called UIMA (Unstructured Information Managment Architecture), which is in active development to support the work of several hundred research and development staff working in areas as diverse as question answering and machine translation. The significant commitment of IBM to SALE development indicates the success of the TEXTRACT concept, and of architectural support for language processing research.

The GATE system has been available free for research since 1996, and as open source software since its second version released in 2002. Bontcheva *et al.* report recent work in upgrading the system to meet challenges posed by research in the semantic web, large-scale digital libraries and machine learning for language analysis.

In the final paper Popov *et al.* present an application that combines several SALE systems, including GATE and Sesame[2], to create a platform for semantic annotation called KIM (Knowledge and Information Management). The paper covers a number of issues relating to architecting scaleable ontology-based Information Extraction.

3 Prognosis

> The principal defining characteristic of NLE work is its objective: to engineer products which deal with natural language and which satisfy the constraints in which they have to operate. This definition may seem tautologous or a statement of the obvious to an engineer practising in another, well established area (e.g. mechanical or civil engineering), but is still a useful reminder to practitioners of software engineering, and it becomes near-revolutionary when applied to natural language processing. This is partly because of what, in our opinion, has been the ethos of most Computational Linguistics research. Such research has concentrated on studying natural languages, just as traditional Linguistics does, but using computers as a tool to model (and, sometimes, verify or falsify) fragments of linguistic theories deemed of particular interest. This is of course a perfectly respectable and useful scientific endeavour, but does not necessarily (or even often) lead to working systems for the general public. (Boguraev, Garigliano, and Tait 1995.)

Working systems for public consumption require qualities of robustness that are unlikely to be achieved at zero cost as part of the normal development of experimental systems in language computation research (Maynard *et al.* 2002). Investing the time and energy necessary to create robust reusable software is not always the right thing to do, of course – sometimes what is needed is a quick hack to explore some simple idea with as little overhead as possible. To conclude that this is always the case is a rather frequent error, however, and this is of particular concern at a time when web-scale challenges to language processing are common.

[2] `http://www.openrdf.org/`

Also problematic for SALE is the fact that it is not always easy to justify the costs of engineered systems when developers of more informal and short-term solutions have been known to make claims for their power and generality that are, shall we say, somewhat optimistic. The fact that the majority of the language processing field has, as we write this in the mid-naughties, used a SALE system of one type or another indicates that this has been a fruitful pursuit. We hope that this collection of papers can contribute to the next wave of research in this area, and to increased reuse, decreased reinvention, and more efficient science and engineering in computation with human language.

Acknowledgements

We are extremely grateful to our editorial committee for their critical analysis on the papers that comprise this issue: Steve Appleby; Steven Bird; Kalina Bontcheva; Chris Brew; Hennie Brugman; Jean Carletta; Walter Daelemans; Robert Dale; David Day; Thierry DeClerck; Marin Dimitrov; Roger Evans; Louise Guthrie; Nancy Ide; Atanas Kiryakov; Diana Maynard; David McDonald; Chris Mellish; Jon Patrick; Thomas Rist; Laurent Romary; Oliviero Stock; Valentin Tablan.

We would also like to thank our many colleagues at Sheffield and Brighton who have helped shape our thinking in the area of SALE, especially Kalina Boncheva, Roger Evans, Diana Maynard, Chris Mellish, Richard Power and Valentin Tablan. Hamish would also like to thank Jon Patrick for his work organising the SEALTS workshop.

During this work Hamish Cunningham was partly supported by EPSRC grant GR/N15764/01 (AKT).

References

Boguraev, B., Garigliano, R. and Tait, J. (1995) Editorial. *Natural Language Engineering*. 1, Part 1.

Bontcheva, K., Tablan, V., Maynard, D. and Cunningham, H. (2004) Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*. This issue.

Brand, S. (1994) *How Buildings Learn*. Penguin, London.

Ferrucci, D. and Lally, A. (2004) UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*. This issue.

Herzog, G., Ndiaye, A., Merten, S., Kirchmann, H., Becker, T. and Poller, P. (2004) Large-Scale Software Integration for Spoken Language and Multimodal Dialog Systems. *Natural Language Engineering*. This issue.

Ide, N. and Romary, L. (2004) Standards for language resources. *Natural Language Engineering*. This issue.

Maynard, D., Tablan, V., Cunningham, H., Ursu, C., Saggion, H., Bontcheva, K. and Wilks, Y. (2002) Architectural Elements of Language Engineering Robustness. *Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data*, **8**(2/3): 257–274.

Mellish, C. and Evans, R. (2004) Implementation Architectures for Natural Language Generation. *Natural Language Engineering*. This issue.

Mellish, C., Scott, D., Cahill, L., Evans, R., Paiva, D. and Reape, M. (2004) A Reference Architecture for Generation Systems. *Natural Language Engineering*. This issue.

Neff, M. S., Byrd, R. J. and Boguraev, B. K. (2004) The Talent System: TEXTRACT Architecture and Data Model. *Natural Language Engineering*. This issue.

Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyanoff, D. and Goranov, M. (2004) KIM – Semantic Annotation Platform. *Natural Language Engineering*. This issue.