A Note on Structured Abstracts

This note briefly examines the role and history of abstracts in scientific and technical papers, and why they are important for software engineering papers.

Roles of abstracts

Although abstracts are now considered to be a standard element of scientific and technological papers, their inclusion is (mostly) a relatively recent development. For most scientific journals, the inclusion of abstracts only dates from the late 1950's (Berkenkotter & Huckin,1995). The main exception to this is the journal *Physical Review*, that adopted the practice of including abstracts in papers in 1920. The later adoption elsewhere may well reflect the rapid expansion of published material in the second half of the twentieth century, leading in turn to the development of abstracting services, with this in turn encouraging researchers to adopt electronic forms of searching for information.

Van der Tol (2001) identifies four main purposes for abstracts:

1. Enabling *selection*, whereby researchers and practitioners use the abstract to help them decide whether an article merits further inspection.
2. Providing *substitution* for the contents of the full document, so that for some readers the information they need is provided without it being necessary to read the full article.
3. Providing an *orientation* function, in the form of a high-level structure that assists with reading all or part of the article.
4. Assisting with *retrieval*, by including information in the abstract that is needed by indexing services, in particular, by highlighting the relevant keywords.

Unfortunately for software engineering too few authors seem to be aware of any of these roles!

The emergence of the evidence-based paradigm has in turn led to increased searching of electronic databases of publications. For this, abstracts perform an important role in terms of *selection* (making the decision about whether or not to include a primary study in a review), along with some elements of *orientation* and *retrieval* when performing data extraction (see the *Guidelines* provided on this site for more details about these activities).

To illustrate the importance of the use of abstracts for *selection*, Table 1 summarises the four-stage process that was followed in selecting the set of studies to be used in the systematic literature review of agile methods reported in (Dybå & Dingsøyr, 2008). Deciding whether or not to include a paper is both an important task for a secondary study as well as potentially time-consuming when it involves having to read the papers themselves. We note that, when describing stage 3, the authors reported that "we found that abstracts were of variable quality" as well as "some abstracts were missing poor and/or misleading, and several gave little indication of what was in the full article'". So this suggests that better abstracts would have considerably reduced the work involved in obtaining and checking the contents of the remaining 270 papers, especially as the final number employed in the study was only 36.

Table 1 Inclusion-exclusion numbers from (Dybå& Dingsøyr, 2008)

Stage No.	*Task*	*No. of studies remaining*
1	Searching journals & conferences using the keyword adopted for the study	1996
2	Excluding studies on the basis of title alone	821
3	Excluding studies on the basis of the abstract	270
4	Appraising the studies on the basis of reading the full paper	36

Abstracts in Software Engineering papers

As in the quotations above, various authors have comments on the poor quality of many software engineering and computing abstracts, perhaps reflecting the relative immaturity of the evidence-based paradigm in this field.

In (Budgen *et al*., 2008) we reported on a study investigating the use of structured abstracts for software engineering papers. A 'structured' abstract is one that is based around the use of a small number of headings, providing the author with guidance on how to structure their summary. This form has been adopted in a number of disciplines in order to increase the completeness and clarity of the information provided in an abstract—and our study demonstrated that this was also true for software engineering papers. Hence we strongly advocate their wider adoption.

In our study we also noted that many of the abstracts used in the study, and which were taken from published studies, were incomplete in some way. So overall, the impression is that software engineering authors tend to give low priority to the task of writing an abstract and few give much consideration to how this might be used. The practice adopted by some conferences of placing a limit on the length of an abstract may also be unhelpful.

A subsequent (and as yet unpublished) study has also demonstrated that the use of structured abstracts helps inexperienced authors to produce better abstracts.

References

Berkenkotter C & Huckin T N (1995). *Genre Knowledge in Disciplinary Communication*, Chapter 2, Erlbaum, N.J.

Budgen D, Kitchenham B A, Charters S C, Turner M, Brereton O P and Linkman S G (2008). "Presenting Software Engineering Results using Structured Abstracts: A Randomised Experiment", *Empirical Software Engineering*, **13**(4), 435-468.

Dybå T & Dingsøyr T (2008). "Empirical studies of agile software development. A systematic review", *Information & Software Technology*, **50,** 833-859.

van der Tol, M (2001), "Abstracts as orientation tools in a modular electronic environment", *Document Design*, **2**(1), 76-88.