How to Make Research Papers in Software Engineering

Kai Koskimies

SoSE Saariselkä seminar 2009

Roles in a research team

- *Manager*: provides basic facilities, funding etc.
- *Idea generator*: provides starting points
- *Idea refiner*: provides idea elaboration skills
- *Technology expert*: provides technical knowledge
- *Architect*: provides general system/tool principles
- *Implementor*: provides design & programming skills
- *Writer*: provides paper writing skills
- *Reviewer*: provides critical viewpoints (internally)
- *Presenter*: provides convincing demos & presentations

Contribution of a paper

- Concept (e.g. a new kind of architectural modeling concept)
- Solution (e.g. general architectural solution, algorithm)
- Method, technique (e.g. a method to derive architecture)
- Tool (e.g. a tool *concept & implementation* to support a method)
- Evaluation of an approach (often case study)
- Finding & interpreting empirical data
- Combination of existing ideas
- … and any combination of these

Paper style

- Empirical science style

  - contribution: empirical data & its interpretation, empirical study of known engineering practices etc., validation of results

- Engineering style

  - contribution: solution to a practical engineering problem & its evaluation

- Mathematics style

  - contribution: algorithm, formal language, correctness, complexity, coverage etc., some path to reality

- … or any combination of these

Antipatterns

- **Empirical data as contribution**
  Present the empirical data (e.g. queries, interviews etc.) you have collected in a summarizing form
  Correction: Decide what is your message, interpret the data, show that your data supports the message

- **Tool as contribution**
  Present your existing tool and show how it can be used to solve a problem
  Correction: Present the problem and solution on an abstract level, and use your tool only in the implementation part, as part of the evaluation

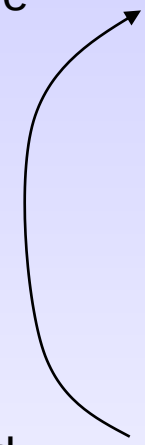
- **Formalism as contribution**
  Show how a certain aspect of software engineering can be formalized
  Correction: Present the formalism as a tool to achieve some advantages or solve some problem in software engineering

Overall structure

Template (engineering style):

Motivating example
- concretizing the problem with an example

Sometimes related work can replace background
(esp. when work is difficult to compare)



Introduction
- context and problem statement, approach

Background
- needed prerequisites, existing concepts used etc

Idea
- beef of the paper, solution on an abstract level

Implementation
- how to realize the idea (e.g. as a tool)

Evaluation
- demonstrate that the idea works, typically a case study

Related work
- comparison with other existing work, highlights contributions

Conclusions
- work seen from bigger perspective, future directions etc.

Acknowledgements
- funding organizaton, comment providers, programmers etc.

References

Title

- is important for selling the paper
- should attract right kind of readers
- can have a catch (e.g. playing with words: ”… REST assured”)
- should not assume special subarea (e.g. ”software architecture” rather than ”architecture”)
- Technique to find a title: collect all the keywords and find all possible sensible combinations of them, connected with auxiliary prepositions, verbs etc. (Method for…, Technique for…, Using X for …, -Driven, -Based, etc.)

Antipattern: Long Title

Introduction

- Remember audience: do not use too much text for painting the scenery
- Introduction is *very important* for acceptance: hook the reader (but do not promise too much, either)
- Template (6 paragraphs):
  - basic context
  - problem description
  - deficiencies with previous approaches
  - basic idea/approach of the paper
  - contributions of the paper
  - textual contents description

Antipattern: Promise Heaven

Background

- Take into account the audience
- The paper should be self-sufficient (with respect to expected audience)
- Try to be as concise and to the point as possible
- Can be given as ”related work”, if that work is more like starting points for this work

Antipattern: Tutorial

Idea

- Don’t overestimate the understanding and guessing capability of the reader
- Use figures to illustrate the main ideas
- Use examples to concretize the concepts
- Use subsections to structure the presentation
- Avoid your own new terms as much as possible
- Don’t create your own world, but a delta for existing world

Antipattern: Explain for your Team

Implementation

- Often necessary to show that the idea has been implemented, part of evaluation
- Architecture-level system descriptions
- Generally interesting technical implementation solutions
- Information about the technical environment
- Use screenshots, whenever appropriate
- Can be rather short

Antipattern: Design Document

Evaluation

- Often in the form of a realistic (industrial) case study
- Example is not a case study
- Make clear what is the thing in the case study you are interested in
- Recall that a case study never *proves* anything
- May provide ”circumstantial evidence” for evaluation
- If possible/sensible, characterize results by quantitative measures (e.g. performance, footprint, lines of code, working time etc.)

Antipattern: Toy Example

Related work

- Comparison of the contribution with existing literature (e.g. one mentioned in Background)
- Template: one paragraph per each existing research direction, with 1-2 statements at the end discussing the differences
- Be careful when discussing about weaknesses of other approaches, be diplomatic
- Perform systematic literature look-up to find relevant references (potential conferences & journals, references in other papers, google)

Antipattern: Crucify Everybody Else

Conclusions

- Idea: provide a high-level perspective on the work (abstract: outside-in, conclusions: inside-out)
- Template:

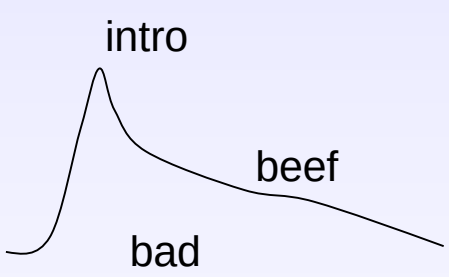
  Rephrasing the main contribution (paragraph)

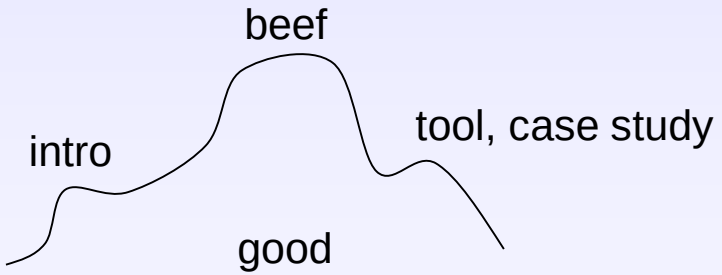
  Significance of the work from larger perspective (paragraph)
- Shortcomings
- Future directions of the work (paragraph)
- New viewpoints but no new results or detailed discussions
- Usually no references

Antipattern: Cut-and-Paste

Focus, focus, focus

- You do not get extra points for extra contents
- You should be able to give the main message of your paper in one or two sentences, stick to that in the paper
- If you are not sure whether a particular ”additional” issue should be discussed or not, probably it should not be discussed
- Make the paper a story, with highlights in the middle





Reviews

- There are many kinds of reviewers (w.r.t. general experience, communication style, specialization area, lazyness); reviews are always subjective
- Try not to take the reviews personally, but merely as input to improve your paper – be professional
- Lack of understanding the paper is not unusual, but that is input for improving the paper, too
- Often difficult to react to comments asking for more explanations, when space is limited
- scope the related work

Research community sociology

- Peer review is the best way to assess papers so far, but very problematic
- There may be self-favouring subcommunities
- Reviewers may have personal agendas and preferences
- Reviewers may be sensitive to trends, hypes and names
- Reviewers are usually conservative
- Reviewers have average ethical standards
- Still, normally reviewers try to do their best