Student Conference 2014

Academic Writing – Part II

Thomas Thüm, Gunter Saake
Line of Thoughts & Cohesion (Roter Faden)

• Maintain cohesive line of thoughts
• Split text into paragraphs
  – connect paragraphs
  – do not jump between topics
• One thought per paragraph
  – Write topic sentence (e.g., first sentence or margin notes, \marginpar)
• Remove unnecessary information
Software product lines promise several benefits compared to individual development [Bass et al., 1998; Pohl et al., 2005]: Due to co-development and systematic reuse, software products can be produced faster, with lower costs, and higher quality. A decreased time to market allows companies to adapt to changed markets and to move into new markets quickly. Especially in embedded systems, in which resources are scarce and hardware is heterogeneous, efficient variants can be tailored to a specific device or use case [Beuche et al., 2004; Tešanović et al., 2004; Pohl et al., 2005; Rosenmüller et al., 2009]. There are many companies that report significant benefits from software product lines. For example, Bass et al. [1998] summarize that, with software product lines, Nokia can produce 30 instead of previously 4 phone models per year; Cummins, Inc. reduced development time for a software for a new diesel engine from one year to one week; Motorola observed a 400% increase in productivity; and so on.
More Topic Sentences

We decided to provide a formalization and proof for both properties, after an initial implementation of our type system for Java. We soon found that our implementation was incomplete: We could not give a guarantee and sometimes generated ill-typed variants because we forgot some

FJ is a minimal functional subset of the Java language for which typing and evaluation are specified formally and proved type-sound with the FJ calculus [8], [40]. It was designed to be compact; its syntax, type judgments and operational semantics fit on a single sheet of paper. FJ strips Java of many advanced features such as interfaces,

So far, we did not discuss the nature of feature annotations and the feature model. As illustrated in our examples in Section 3, we are interested in reachability conditions like the following sentence ‘whenever code fragment a is present, then also code fragment b is present’ based on their annotations and additional constraints of the feature model. (We use the metavariables a and b to refer to arbitrary annotatable code fragments.) Reachability is necessary, for example, to check whether a method invocation in code fragment a can always reference a method declaration in
Coherence

<table>
<thead>
<tr>
<th>Intro</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The point</strong> (best)</td>
<td>...or here (ok)</td>
</tr>
</tbody>
</table>
Coherence on a Large Scale
Say what you say before you say it

• Explain the structure of the text
• Pick up the readers, guide them, prepare them
• Connect chapters and sections
• Support readers in skimming the paper („Querlesen“) or interruptions

Benefits of AST representation

The AST representation has three main benefits: improved expressiveness, easier use, and opportunities for extensions.

First, we improve expressiveness, since we can classify more annotations as dis-

7. IMPLEMENTATION & CASE STUDIES

In the previous sections, we have designed and formalized a product-line-aware type system. To demonstrate its practicality, we implemented it in our tool CIDE and performed a series of case studies to evaluate performance and whether we can actually find type errors in existing product lines.

7.1 Implementation
Avoid mere description

Explain what you are doing and **why**

We implemented a type system in our tool CIDE and performed a series of case studies.

vs.

To demonstrate practicality, we implemented a type system in our tool CIDE and performed a series of case studies.
Self Contained

- You are an expert on the topic – your readers are probably not

- Provide all necessary background information for understanding your work
  - Be concise
  - Provide references for further details
  - A reference does not replace explaining necessary background

- Know your audience
Stating the Contribution

• Make contribution crystal clear
• Don’t be shy
• Be very specific: “we contribute”

The main innovation of this chapter is our revised type system for CFJ. The type system known from literature can be simplified due to redundant premises at the some typing rules. A smaller contribution is that we give some new and adapted examples of FJ programs and CFJ product lines.

Perspective, Goals, and Contributions. In this paper, we examine functional aspects in the light of AOR. Function evaluation imposes a fixed weaving order, but also a fixed refactoring order. That is, we cannot factor out aspect A.
Stating the Contribution (Example)

say that they are misused. To improve the situation, we make the following contributions:

- We analyze object-oriented modifiers used in FOP and identify several shortcomings that lead to a limited expressiveness of feature-oriented languages, undefined program behaviors, and inadvertent type errors.
- We explore the design space of feature-oriented access control mechanisms and propose three concrete access modifiers.
- We present an orthogonal access modifier model, which integrates common object-oriented modifiers with our novel feature-oriented modifiers.
- We offer an implementation of the proposed modifiers on top of the fully-fledged feature-oriented compiler Fuji.
- We analyze ten feature-oriented programs and demonstrate that there is a potential for feature-oriented modifiers in practical FOP.

Especially, the last two contributions are novel compared to an earlier version of the paper presented at FOSD’09 [11].
Overclaims

• Be careful with overclaims that you cannot prove
• Narrow it down to your actual contribution, be precise

Our approach provides reliable high-performance data access

Existing database systems are slow and do not scale
Referencing Publications

• Reference ideas and prior work

• Always reference used or adopted figures
  • e.g., “Figure 2: Feature model of Berkeley DB, adopted from [2]”
  • Copyright can be an issue

• NEVER copy and paste text from papers or websites
  • Paraphrase ideas
  • Also be careful when copying from yourself
  • More ethics on this later...
Citation Style

• Direct quotations are not common, except for definitions
• Typically use quotation at the end of a sentence
  • „We formally extend Featherweight Java (FJ) – a Java subset proved type-sound using a concise calculus [41].“
  • „Without loss of generality, we focus on FODA-style feature models [12, 43], because ...“
  • „Parnas suggests dividing programs according to concerns instead of purely technical considerations [13].“
• Do not use reference as subject; avoid “see”
  • “[13] shows additional statistics” (bad)
  • “see [13] for additional statistics” (bad)
  • “In [13], Hu et al. show additional statistics” (borderline)
  • “Hu et al. presented additional statistics [13]” (better)
Citing Own Work

• Make clear when referencing own work
  • “This problem was studied earlier, but in a less general setting [2,3,5].” (bad)
  • “We studied this problem earlier [2,3,6], but in a less general setting.” (better)
  • “In prior work, we studied this problem in a less general setting [2,3,6]” (better)
Reference Style

• In papers
  – Typically numbered references are used [1], [2]
  – Page numbers omitted

• In a thesis
  – rather use abbreviations [ATG09] or better author-year style [Apel and Saake, 2006] (for Latex see package natbib)
  – Provide page numbers for books [S99, pp. 55-59]

• Different researchers prefer different styles. Ask advisers when writing a thesis. Check formatting guidelines of publishers.
Formatting Bibliographies

• References must include
  – Name of authors
  – Title
  – Where published
    • Journal Article: Journal & Volume & Edition & Pages
    • Conference Paper: Conference & (Series and volume) & Pages & Publisher
    • Book: Publisher
    • Technical Report: Number & Department & University
  – Year
• ISBN, ISSN, DOI, location, date, editors and others are optional and usually not included (if you include them be consistent and include them for all references)
Clean Your Bibliography

• An inconsistent/incomplete bibliography makes a bad impression, check consistency early on

• When importing bibtex entries, check for style and consistency

• Typical problems
  – Information missing (no publisher, no pages)
  – Inconsistent upper and lower case
    • Classbox/j: Controlling the scope of change in java
    • Aspect-Oriented Programming
  – Inconsistent names for conferences/journals, inconsistent abbrev.
    • Proc. Int’l Conf. Software Engineering (ICSE)
    • ICSE’08: Proceedings of the 30th International Conference on Software Engineering
    • Proceedings International Conference on Software Engineering
Tip for BibTeX Users: Constants for Consistency

@String{OOPSLA = "Proc.\ Int'l Conf.\ Object-Oriented Programming, Systems, Languages and Applications (OOPSLA)"}
@String{ICSE = "Proc.\ Int'l Conf.\ Software Engineering (ICSE)"}
@String{ECOOP = "Proc.\ Europ.\ Conf.\ Object-Oriented Programming (ECOOP)"}
@String{TSE = "IEEE Transactions on Software Engineering (TSE)"}
@String{CACM = "Communications of the ACM"}
@String{ViSPLE = "Proc.\ SPLC Workshop on Visualization in Software Product Line Engineering (ViSPLE)"}
@String{LNCS = "Lecture Notes in Computer Science"}
@String{GI = "Gesellschaft f\"ur Informatik (GI)"}
@String{ACM = "ACM Press"}
@String{Springer="Springer-Verlag"}

@inproceedings{LBL:ICSE06,
    author = {Jia Liu and Don Batory and Christian Lengauer},
    title = {Feature Oriented Refactoring of Legacy Applications},
    booktitle = ICSE, publisher=ACM, address=ACMAAddr, year = 2006,
    isbn = {1-59593-375-1}, pages = {112--121} }

10.04.2014  Student Conference
Examples


• Benjamin C. Pierce. Types and Programming Languages. MIT Press, 2002.

**No Publisher?**

- Sometimes proceedings of workshops are published in technical reports by companies or universities

- When papers of a workshop are only published online, provide URL
Referencing URLs

• Don’t

• Consider using a footnote instead
• If you really must reference an URL, provide date of access
• If you can provide authors
• Reference specific version of wikis or other pages that keep a history
  – http://lampiro.googlecode.com/svn/!svn/bc/30/trunk/
Literature
Bugs in Writing I

• Do not write passive! Use the active voice.
  – My first visit to Boston will always be remembered ✗ me.
  – I shall always remember my first visit to Boston ✓.

• Like vs. Such as
  – We do not support advanced programming languages, such as Java.
  – We do not support advanced programming languages like most embedded systems.
  – Different meanings!
Bugs in Writing II

• Where to use „only“
  – I only make programs.
    • ... I do not execute, benchmark, debug them.
  – I make only programs.
    • ...I do not make cake, cars, scripts.
  – Only changes the meaning of the preceding word!

• Undefined This (always use a noun after this, these, that, some)
  – Norbert asked Janet for help in statistics to write about it in the paper. This was a good idea.
    • Asking Janet, because she is an expert?
    • Putting statistics in the paper, because it improves the paper?
Bugs in Writing III

• Which vs. That
  – The program that is corrupting the computer must be removed.
  – The program, which is corrupting the computer, must be removed.

• Which program must be removed?
  – The one that is corrupting the computer!
  – That picks out a single program from a set. It identifies objects!

• Which gives us only additional information.
  – The program must be removed. Oh and it is corrupting the computer.
Bugs in Writing IV

• Abbreviations
  – Disrupts the reading flow. Use carefully!
  – Introduce it: Software product lines (SPLs) are ...
  – Do not always create abbreviations from first letters.

• Standard latin abbreviations:
  – E.g., (exempli gratia) stands for: for example
  – i.e., (id est) stands for: that is
  – Et al. (et alia) stands for: and others

• If you start with e.g., or for example, do not write etc. at the end
• **Reason is because**
  – Redundant phrase
  – Reason is that ..., the reasony why, or because

• **Deduce vs Infer**
  – Direction of reasoning
  – Deduction: from general to specific; from population to individuals
    • All men like beer. We deduce that Norbert likes beer.
  – Inference: from specific to general; from sample to population
    • Our admins use Linux. All admins in the world use Linux.
Bugs in Writing VI

• In order to
  – Long and verbose
  – Instead: to, so as to

• The fact that
  – Never use the fact!
  – Just use that

• Utilize
  – Just do not use it. Use „use“ instead. 😊
Bugs in Writing VII

• **Whether vs. If**
  – Use “whether“ when you can say: whether or not (and usually omit „or not“)
    • I don’t know if I can hold the deadline
    • I don’t know whether I can hold the deadline
  – Use „If“ when you are writing a constraint
    • I will submit the paper at the deadline if it is in a good shape.

• **Neither … nor (opposite of either or)**
  – Use „nor“ with „neither“ and „or“ with „either“
Bugs in Writing VIII

• Will likely be (incorrect)
  – Probably will be
  – Is likely to be

• Firstly, secondly, ... (incorrect)
  – Use First, Second, ...

• Since vs. Because (be careful)
  – Since usually refers to time: Since one week, we ...
  – Because refers to a causal relationship: He is tall, because...
  – However, since is often used today as a synonym for because
Bugs in Writing IX

• Don’t always use problem
  – There are no problems... only challenges! 😊

• Better, Best, Worst
  – Vague and ambiguous
  – Use words that specify the measure of goodness
    • Accurate
    • Cheaper
    • Unexpected behavior
    • Effective in terms of energy consumption
    • Faster
Bugs in Writing X

• Do you know how to write:
  – Dates and Times of Day?
  – Percentages? (as symbol or word)
  – Numbers (when numbers, when word)
  – US vs. British spelling
  – Citations
  – Hyphens
  – Pronouns
  – Repeated prepositions
  – Etc.
General Principles I

• Put statements in positive form
  – The algorithm was not very often correct.
  – The algorithm was usually wrong.

• Place negative in positive in opposition
  – Ask not what your country can do for you – ask what you can do for your country.

• Do not use conditionals (sounds irresolute)
  – Applications can make a good impression when...
  – Applications will make a good impression when...
General Principles II

• Use definite, specific, concrete language
  – Unfavorable weather vs. It rained
  – Algorithm performed better vs. Algorithm was faster

• Omit needles words
  – The question is to whether vs. Whether
  – Used for performance purposes vs. Used for performance
  – In an accurace manner vs. Accurately
  – The reason is why vs. Because
General Principles III

• Keep related words together
  – You can call your mother in London and tell her all about George’s taking you out to dinner for just two dollars.
  – For just two dollars, you can call your mother in London and tell her all about George’s taking you out to dinner.

• Place the emphatic words of a sentence at the end
  – This steel is principally used for making razors, because of its hardness.
  – Because of its hardness, this steel is used principally for making razors.