Source Review

Van Deursen, Arie and Kuipers, Tobias

Identifying objects using cluster and concept analysis

```
@inproceedings{vanDeursen:1999:IOU:302405.302629,
 author = {van Deursen, Arie and Kuipers, Tobias},
 title = {Identifying objects using cluster and concept analysis},
 booktitle = {Proceedings of the 21st international conference on Software engineering},
 series = {ICSE '99},
 year = {1999},
 pages = {246--255},
 publisher = {ACM},
 address = {New York, NY, USA},
}
```

Research Area: Software Engineering

Summary

The authors evaluate agglomerative hierarchical clustering and concept analysis for assisting in the derivation of object-oriented class structure from a legacy COBOL application.

The paper refers to other uses of clustering techniques in the extraction of class structure. The primary extension to this other work is the decomposition of large records during clustering, motivated by the authors' observation that the legacy system they are migrating contains bloated data structures that should be decomposed into smaller, more elegant classes. Actual use of COBOL record fields within COBOL programs is clustered, with distance between fields measured by counting the number of programs that contain both fields. A similar approach is used for concept analysis. Both approaches are evaluated via discussion with the originators of the code, and by comparison with an independent re-architecture by experts. Then, the approaches are compared, with concept analysis being preferred as they feel it better maps to the hierarchical object-oriented design, with less filtering of the codebase required before analysis. The authors also note that they managed to extract the same core classes as the experts, though they did not extract all of the same classes.

Comments

This sounds like a useful way to analyze code, and likely quite generalizable. Field-accessors are part of many languages, and different lexers could be developed to support these languages. Additionally, a similar approach could be used on code that is already object-oriented to decompose classes that have become too large and unwieldy over time.

Though concept analysis was explored and ultimately preferred, there almost no discussion of how it has been used previously, other than the citation of four other papers that explored its use in "analyzing the modular structure of legacy software". It felt to me that I should have seen some mention of at least one of these papers in the related works section.

In the introduction, some problems with semi-automated migration approaches are enumerated. Two of three are related to the fact that many systems are heterogeneous code-bases, containing many languages, persistence technologies, and so on. The authors then describe an approach that deals with one programming language. I understand that they are attempting to motivate the necessity of ad hoc approaches, and thus their exploration within the context of many other similar analyses, but this section feels disconnected from the rest of the paper.

I think it's good that the authors validated their exploration with programmers who worked on the code. This will help take some of the guesswork out of the ideal structure of the migrated application.

I haven't heard of concept analysis before, but it sounds interesting. It seems like a good fit for software-engineering analysis in general.

- Logan Gilmour