Topic Modelling of Climate Model Source Code

Abram Hindle
University of California, Davis
Davis, CA
ah@softwareprocess.es

1. INTRODUCTION

Eric Evans, in his book Domain Driven Design [1], argues that all stakeholders should try to share a similar vocabulary and lexicon when developing software. The idea is that if developers use the same agreed upon terminology that other stakeholders use, such as domain experts in the field being modelled, that serious design issues and domain-specific defects in software can be fixed or avoided very early during development.

Climate modelling is a very complex and complicated domain. To a software practitioner unfamiliar with climate modelling some of the entities found in the code might be unclear. Optimizations and common representations of the data used in climate modelling are probably unknown the practitioner.

An example of a semantic failure that is avoidable by using a shared vocabulary is the use of arrays of numbers. These arrays are effectively untyped. Does one understand the assumption of the data type, is it a count of elements, is it a unit measurement such as temperature, what is the unit of the array, is it time-series data, is it a distribution? A software practitioner might not know any of this unless they are familiar with the domain. They might understand the purpose if they knew the terminology to describe the function, and the terms used to name the array. But how can a practitioner gain enough domain knowledge such that their debugging can be fruitful?

1.1 Proposal

We propose a potential solution: a word network, or dictionary, of terms pulled from existing climate modelling systems. We propose associating source code terms, documentation terms and literature terms together in order to form unifying and shared concepts. For instance if `temp` appears in an identifier, how often does it relate to temperature versus temporary value. Perhaps in many climate modelling code bases `tmp` means temporary while `temp` means temperature. If a practitioner can look up the possible meanings of identifiers and identifier fragments they can avoid making dangerous assumptions.

1.2 Immediate Application

Our first step would be to investigate topics extracted from the source code and documentation of Nemo[1] and GISS GCM ModelE[2] by by techniques such as Latent Semantic Indexing (LSI) or Latent Dirichlet Allocation (LDA). These tools are unsupervised methods of teasing out underlying independent word distributions from documents, such as source code [2]. We would then investigate the topics extracted and identify climate-modelling specific and software engineering relevant topics.

[1] `http://www.nemo-ocean.eu/`
[2] `http://www.giss.nasa.gov/tools/modelE/`

We hope to find topics belonging to the climate modelling domain and the software engineering domain. These topics will be sets of words and can provide a reasonable starting point for any kind of lexicon elicitation or grooming.

1.2.1 Potential Applications

Future potential tools include:

Concept Thesaurus would allow plug-ins in Eclipse and other IDEs to access a thesaurus or ontology of concepts. One could right click on a word or token and see the relevant annotations of that concept.

Lexicon Enforcer would be a tool to ensure that when new concepts are added, they are documented (or ignored). The lexicon enforcer could be used to help developers and modellers keep a consistent and tight lexicon without a lot of semantic drift.

Code idiom look-up by leveraging code cloning and code query technology, one could check how often a certain idiom was being used in contexts such as: correct climate data, smoothing, model assumptions. A sample usage would be to highlight code in the IDE that was unclear to the practitioner and see where it re-occurs across potentially multiple projects. One could also allow the annotation of patterns. This is valuable in languages like C, Fortran and Java where sufficient efficient abstractions might not be available.

1.3 Conclusion

In conclusion we propose to extract and analyze topics from the source code, documentation and change history of climate modelling software that is available to us. We plan to interpret these results and relate terms to each other via topics.

We hope to find software engineering topics as well as domain specific (climate modelling) topics that can be used to relate documents and terms to actual topics of importance.

2. REFERENCES

[1] E. Evans. *Domain-Driven Design: Tacking Complexity In the Heart of Software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[2] A. Kuhn, S. Ducasse, and T. Girba. Semantic clustering: Identifying topics in source code. *Information and Software Technology*, 49(3):230–243, March 2007.