3-9-2007

Citation Analysis using Formal Concept Analysis: A case study in Software Engineering

T. Tilley
University of Wollongong

P Eklund
University of Wollongong, peklund@uow.edu.au

Follow this and additional works at: <https://ro.uow.edu.au/infopapers>

Part of the [Physical Sciences and Mathematics Commons](#)

Recommended Citation

Tilley, T. and Eklund, P: Citation Analysis using Formal Concept Analysis: A case study in Software Engineering 2007.
<https://ro.uow.edu.au/infopapers/606>

Citation Analysis using Formal Concept Analysis: A case study in Software Engineering

Abstract

In this paper Formal Concept Analysis (FCA) is used as a means to analyse a field of research using published academic papers as its input. In particular, results are presented based on a case study of 47 academic papers in a scientific field of study. The analysis includes inferences about the field of study based on the domain background knowledge derived from the ISO12207 software engineering standard. Additionally, a number of alternative classifications based on the target application language and the reported application size are introduced. FCA reveals useful insights about the nature of the subject matter: identifying fruitful areas of research as well as producing details about characteristics of the community under examination.

Disciplines

Physical Sciences and Mathematics

Publication Details

This paper was originally published as Thomas, T, and Eklund, P, Citation Analysis using Formal Concept Analysis: A case study in Software Engineering, 18th International Conference on Database and Expert Systems Applications DEXA 2007, 3-7 Sept, 545-550.

Citation Analysis using Formal Concept Analysis: A case study in Software Engineering

Thomas Tilley and Peter Eklund

School of Information Systems and Technology
The University of Wollongong, NSW 2522, Australia
atettilley@yahoo.com.au, peklund@uow.edu.au

Abstract *In this paper Formal Concept Analysis (FCA) is used as a means to analyse a field of research using published academic papers as its input. In particular, results are presented based on a case study of 47 academic papers in a scientific field of study. The analysis includes inferences about the field of study based on the domain background knowledge derived from the ISO12207 software engineering standard. Additionally, a number of alternative classifications based on the target application language and the reported application size are introduced. FCA reveals useful insights about the nature of the subject matter: identifying fruitful areas of research as well as producing details about characteristics of the community under examination.*

1 Introduction

Formal Concept Analysis (FCA) [9] can be used to provide a structural view over contributions in the field of study. In particular it can be used to examine the relationships between articles based on their content as well as by characterizing them through the presence of citations. Scientific papers in the designated field of study were collected and attributes assigned to them on the basis of their content, citations and keywords. Background knowledge particular to the field is also introduced. This provides implicit content categories in the field of study. A similar idea is used by So [20] where collected conference papers are split into 13 interest areas or conference tracks. In our treatment we use the categories of the ISO12207 software engineering standard as a natural division of subject material covered by the papers within the field. Papers that are cited but do not fit into the ISO12207 software engineering standard were excluded from our analysis.

In this paper, we present similar ideas and content to an earlier paper on the application of FCA to citation analysis. Our intention in presenting at ACKE'07 is to present a much larger coverage of ideas that supercedes both this paper and the material published by the authors and others earlier in [21]. A more extensive contribution has already been prepared as a journal paper and therefore ACKE'07 provides a forum to re-open the ideas and issues involved in citation analysis and FCA.

The structure of this paper is as follows. Being familiar with FCA, preliminaries describing the basic

FCA results are omitted for the ACKE'07 audience. However, to fully understand the case example, some background on software engineering (SE) is introduced. Specific data mining results based on the relationship between the field of study and background knowledge in software engineering are also presented.

2 Formal Concept Analysis in Software Engineering

The development of software has traditionally been described by life-cycle models. These models grew out of a need to more effectively understand and manage the software engineering process. Royce [16] proposed the classic waterfall model which consists of 7 *steps* or *phases* that proceed in a linear fashion: System Requirements, Software Requirements, Analysis, Program Design, Coding, Testing, and Operations. A number of other life-cycle models exist and the most appropriate model to use for a given project depends on a number of factors including the type of project, the style of the developers and the organisational maturity of both the developers and customer.

An alternative to the classic life-cycle approaches is to use a meta-model that defines common software engineering activities independently of a particular life-cycle model. Developers can then choose the most-appropriate life-cycle for their project and the activities can be mapped onto the chosen model. The ISO12207 Software Engineering Standard [11] describes such a meta-model for software engineering life-cycle. The standard includes 13 activities that can be mapped onto a chosen life-cycle model. The first of the activities is related to starting the methodology, another 4 are system related and the remaining 8 relate to the software itself. The 13 activities are: (1) Process implementation; (2) System requirements analysis; (3) System architectural design; (4) Software requirements analysis; (5) Software architectural design; (6) Software detailed design; (7) Software coding and testing; (8) Software integration; (9) Software qualification testing; (10) System integration; (11) System qualification testing; (12) Software installation; (13) Software acceptance support.

The ISO standard notes that “these activities and tasks may overlap or interact and may be performed

iteratively or recursively”. In addition to the 8 software related activities defined above an understanding of software maintenance is also required. The IEEE Standard Glossary of Software Engineering Terminology defines software maintenance as “The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment [10]”. The process requires iteration through some or all of the previously defined activities and in terms of the water-fall model it could be thought of as a feedback loop to previous stages.

In our case the ISO12207 Standard is used as a generic background knowledge model for our case study analysis because it provides structured meta-categories. These categories can be used as attributes for describing the 47 papers in the survey which report software engineering applications for FCA. The study is circumscribed because when constrained in this way there are relatively few papers and the results of the analysis can be verified by observation. This facilitates the presentation of the case study given the constrained format.

ResearchIndex (formerly known as “CiteSeer”) is a scientific digital library project that was originally a demonstration site for the NEC Research Institute’s CiteSeer software. CiteSeer was designed to automatically gather and index citations from papers published on the World Wide Web (WWW). Citation indexing links articles based on a bibliography of cited articles or references within one paper being matched against the titles of other documents within the database. This process is automated in ResearchIndex and both papers cited within a document and papers citing a document can be retrieved. The collection of 47 papers for our case study where found in the ResearchIndex and built around a set of core papers chosen as initial seed objects. Our own study had identified the obvious contributions to the field, for example Snelting [19]. This paper provided an overview of software re-engineering based on concept lattices. Using this paper as a starting point, more papers were added to the collection via the ResearchIndex. Only papers that were accessible to ResearchIndex, i.e. contained within the on-line index, were used as candidate papers for the case study. While this means that the collection may be incomplete, we argue that it is comprehensive and captures the majority of scientific papers on this subject. Being reasonably familiar with the field, even where some papers are missing, we can verify that the major scientific ideas within the field of study are contained within the 47 papers used in the case study. For the presentation of our case study, familiarity is an advantage but we concede it could also be a disadvantage of the technique in a field that was unfamiliar.

A first classification of the papers that considers the software-related ISO12207 activities as attributes appears in Table 1. The intention was not to classify a

paper exclusively with a single activity but to record all the activities supported by the approach presented in the paper. Note also that “coding and testing” appears as a single activity in the ISO12207 standard but has been broken down into two separate attributes for the classification context. This context actually represents a sub-context of the total set of survey attributes and therefore represents a conceptual scale which captures the ISO12207 activities.

2.1 ISO12207 Categorisation

	Requirements Analysis	Architectural Design	Detailed Design	Coding Testing	Integration	Qualification Testing Installation	Acceptance Support	Software Maintenance
Ammons03debugging				x				x
Andelfinger97diskursive	x							
Arevalo03understanding-a		x						x
Arevalo03understanding-b		x						x
Ball99concept				x				x
Boettger01reconciling	x							
Bojic00reverse	x	x						x
Canfora99case			x					x
Dekel02applications			x					x
Duvel98identifying	x	x						
Duvel99enhancing	x	x						
Duvel00bridging	x	x						
Eisenbarth01aiding		x						x
Eisenbarth01feature		x						x
Eisenbarth03locating		x						x
Fischer98specification		x	x					
Funk95algorithms			x					x
Godin93building			x					x
Godin95applying			x					x
Godin98design			x					x
Huchard99from			x					x
Huchard02when			x					x
Krone94inference			x					x
Kuipers00types			x					x
Leblanc99environment			x					x
Lindig95concept			x					
Lindig97assessing			x					x
Richards02assisting	x							
Richards02controlled	x							
Richards02recocase	x							
Richards02representing	x							
Richards02using	x							
Sahraoui97applying			x					x
Schupp02right			x					x
Siff97identifying			x					x
Snelting96reengineering			x					x
Snelting98reengineering			x					x
Snelting98concept			x					x
Snelting99reengineering			x					x
Snelting00software			x					x
Snelting00understanding			x					x
Streckenbach99understanding			x					x
Tilley03software	x	x						
Tilley03towards		x	x					
Tonella99object			x					x
Tonella01concept			x					x
vanDeursen98identifying			x					x

Table 1. Context considering the 47 papers in the survey as objects and the ISO SE activities as attributes.

References to papers included in the survey use the naming format adopted by the ResearchIndex digital

library [1]. For example Krone and Snelting’s paper entitled “On The Inference of Configuration Structures from Source Code” and published in 1994 would appear as *Krone94inference* [12]. The concept lattice corresponding to the content in Table 1 appears in Figure 2 and it can be seen that 27 out of the 47 papers in total describe applications to both *software maintenance* and *detailed design*. These papers mostly report the use of FCA to identify class candidates in legacy code or the maintenance of class hierarchies. Considering the theory behind the sub/superconcept ordering within a formal concept lattice this is an obvious inference.

An emerging body of literature related to *requirements analysis* can also be seen with 12 of the 47 papers reporting application in this area. Across the total set of survey papers it is also noteworthy that there are only 2 papers describing applications to *testing* and none of the collection explicitly report application to *software integration*, *qualification testing*, *installation*, *acceptance support* or *coding*.

Thirty-three of the survey papers have been classified as *Software Maintenance* applications. Additionally, the work described in *Eisenbarth01aiding* [5], *Eisenbarth01feature* [6] and *Eisenbarth01aiding* [7] has also been categorised as *Architectural Design*. The papers *Ammons03debugging* [2] and *Ball99concept* [3] incorporate *Testing* and these are the only 2 papers in the survey to address this activity. *Bojic00reverse* [4] discusses an application to round-trip engineering and recovering UML use-cases. This work has therefore also been included under the *Requirements Analysis* category even though it is concerned with re-engineering existing systems.

Figure 1. The ISO12207 categorisation diagram from Figure 2 showing the paper names.

While the bulk of the papers in the case study are applications to late phase SE activities, 14 of the 47 papers are concerned with early-phase SE. The techniques described in 27 of the 33 *Software Maintenance* papers also necessitate design reviews or at least pro-

posed changes to the design of legacy systems: as such they have also been categorised under *Detailed Design*.

Figure 2. The Concept Lattice corresponding to the context in Table 1. The objects are the 47 papers included in the survey while the attributes are the activities defined in the ISO12207 standard.

Note that only object counts are shown on the diagram and the node shading indicates the distribution of objects. In addition to the attributes appearing in the context shown in Table 1, there were 133 attributes used in total to categorise the papers in the survey. These attributes included the names of the authors, citations of other papers in the survey, the year of publication, inputs, outputs, target application programming languages (e.g. C++, Java) and the “size” of any reported application target.

The context in Table 2 represents the application of the approach described within a paper to a particular language. The attributes here are the programming languages: C, C++, COBOL, FORTRAN, Java, Modula-2, Smalltalk, and the design or specification languages: OMT, UML and Z. Both procedural and object-oriented languages are represented. The attribute values record the size of any reported target application in KLOC (“thousand Lines Of Code”) — for example, 106 KLOC represents an application containing 106,000 lines of source code. KLOC is also sometimes referred to as SKLOC (Source Thousand Lines Of Code) and is a metric that is often reported to indicate project size in software engineering. Where a paper contains multiple examples for the same language only the largest application size is shown.

While there is some debate about the usefulness of size-oriented metrics like KLOC [15] it does give a raw indication of application size. Within the set of survey papers it may also be indicative of tool support and helps isolate identical examples. It is also interesting to note that where a non-zero value repeats in the context it typically refers to the same example being reported in a number of papers. For example, the same 1.6 KLOC C application appears in the papers *Funk95algorithms* [8], *Krone94inference*, *Snelting96reengineering* [17], *Snelting98concept* [18]

The largest application in the survey describes the analysis of a 1,200 KLOC semiconductor testing tool written in C. The work by Eisenbarth, Koschke and Simon in *Eisenbarth03locating* is an order of magnitude larger than any of the other examples and demonstrates that FCA-based software analysis tools are capable of handling real-world projects.

From the initial ISO12207 categorisation of the papers it is evident that the majority describe software maintenance and re-engineering applications. None of the papers support *acceptance support*, *integration*, *coding*, *installation* or *qualification testing* and the papers by Ball, and Ammons et al. stand out as the only *testing* related applications. This identifies promising gaps in the literature that can, if addressed, result in contributions to new knowledge.

A more comprehensive analysis of this example is extended in the unpublished journal paper which also illustrates patterns of collaboration from the research literature. That paper will follow from participation in ACKE'07.

3 Conclusion

The paper has provided an overview of FCA applications to software engineering via a case study. Our paper contributes by presenting a novel way of performing a literature survey of 47 academic papers reporting software engineering applications for FCA using FCA. The paper provides an overview of the research papers using conceptual scales based on ISO12207 categorisation, target language and target application size. While these attributes are specific to the domain of software engineering other attributes which are common to all academic literature can often be found. This ACKE'07 paper is an abbreviated version of a more comprehensive analysis available on request.

References

- [1] CiteSeer: The NEC research institute scientific literature digital library, 2002.
- [2] G. Ammons, D. Mandelin, R. Bodik and J.R. Larus. Debugging temporal specifications with concept analysis. In *Proc. of the Conf. on Programming Language Design and Implementation PLDI'03*. ACM, 2003.
- [3] T. Ball. The concept of dynamic analysis. In *Proc. of ACM SIGSOFT Sym. on the Foundations of Software Engineering*, pages 216–234, 1999.
- [4] D. Bojic and D. Velasevic. Reverse engineering of use case realizations in UML. In *Symposium on Applied Computing - SAC2000*. ACM, 2000.
- [5] T. Eisenbarth, R. Koschke and D. Simon. Aiding program comprehension by static and dynamic feature analysis. In *Proceedings of ICSM2001 - The International Conference on Software Maintenance*, pages 602–611. IEEE Computer Society Press, 2001.
- [6] T. Eisenbarth, R. Koschke and D. Simon. Feature-driven program understanding using concept analysis of execution traces. In *9th Int'l Workshop on Program Comprehension*, pages 300–309. IEEE, 2001.
- [7] T. Eisenbarth, R. Koschke and D. Simon. Locating features in source code. *IEEE Trans on Software Engineering*, Volume 29, Number 3, pages 195–209, 2003.
- [8] P. Funk, A. Lewien and G. Snelting. Algorithms for concept lattice decomposition and their applications. Technical Report 95-09, TU Braunschweig, 1995.
- [9] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.
- [10] IEEE. *IEEE Std 610.12-1990 — IEEE Standard Glossary of Software Engineering Terminology*. IEEE, New York, September 1990.
- [11] IEEE. *IEEE/EIA 12207.0-1996 — Standard for Information Technology – Software life cycle processes*. IEEE, New York, March 1998.
- [12] Maren Krone and Gregor Snelting. On the inference of configuration structures from source code. In *Proceedings of the International Conference on Software Engineering (ICSE 1994)*, pages 49–57, 1994.
- [13] T. Kuipers and L. Moonen. Types and concept analysis for legacy systems. Technical Report SEN-R0017, Centrum voor Wiskunde en Informatica, July 2000.
- [14] C. Lindig and G. Snelting. Assessing modular structure of legacy code based on mathematical concept analysis. In *Proc. of the Int. Conf. on Software Engineering (ICSE 97)*, pages 349–359, Boston, 1997.
- [15] R. Pressman. *Software Engineering: a practitioner's approach*. McGraw-Hill, Singapore, 3rd edition, 1992.
- [16] W. W. Royce. Managing the development of large software systems. In R.H. Thayer (editor), *Tutorial: Software Engineering Project Management*, pages 118–127. IEEE Computer Society, Washington, 1987. Originally published in Proceedings of WESCON'87.
- [17] G. Snelting. Reengineering of configurations based on mathematical concept analysis. *ACM Transactions on Software Engineering and Methodology*, Volume 5, Number 2, pages 146–189, April 1996.
- [18] G. Snelting. Concept analysis: a new framework for program understanding. In *SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools & Engineering (PASTE)*, pages 1–10, 1998.
- [19] G. Snelting. Software reengineering based on concept lattices. In *Proceedings 4th European Conference on Software Maintenance and Reengineering*, pages 3–12. IEEE, 2000.
- [20] Clement Y. K. So. Citation ranking versus expert judgment in evaluating communication scholars: Effects of research specialty size and individual prominence. *Scientometrics*, Volume 41(3), pages 325–333, 1998.
- [21] T. Tilley, R. Cole, P. Becker and P. Eklund. *FCA — The state of the art*, Chapter A Survey of Formal Concept Analysis Support for Software Engineering Activities. LNCS3626. Springer-Verlag, 2005.
- [22] A. van Deursen and T. Kuipers. Identifying objects using cluster and concept analysis. In *Proceedings of the 21st International Conference on Software Engineering, ICSE-99*, pages 246–255. ACM, 1999.