# Understanding the Structure of Agile Software Development Using Text Analytics: A Preliminary Analysis

*Submission Type: Full Paper*

**Sridhar Nerur**
University of Texas at Arlington
snerur@uta.edu

**VenuGopal Balijepally**
Oakland University
balijepa@oakland.edu

## Abstract

The tenets of agile software development (ASD) were formulated over fifteen years ago. Since then, a number of methods and best practices have emerged, which, in turn, have spawned many research ideas. This study attempts to chronicle the evolution of thought in agile software development by using text analytics, an approach that is becoming invaluable in our efforts to understand unstructured text. Specifically, we use text analytics to unravel latent semantic relationships within the agile domain in order to get a sense of where we started, where we are today, and what to anticipate in the future.

**Keywords**

Agile development, text analytics, document co-citation analysis, intellectual structure

## Introduction

The field of software development has undergone major changes since the articulation of the Agile Manifesto in 2001 (AgileAlliance 2001). Quite understandably, confusion was rife in the initial years, with questions being raised about the efficacy and value of agile software development (ASD) (Boehm 2002). ASD's emphasis on a collaborative approach that relies on adaptive planning, short cycles of development, and continual interaction with a customer was a departure from the orthodoxy that had been around for several decades (Cockburn and Highsmith 2001; Highsmith and Cockburn 2001; Nerur et al. 2005). It is increasingly apparent that the days of skepticism are behind us, and a growing number of organizations have either blended in agile concepts with their traditional practices or have made agile their dominant software development strategy (Ambler 2013).

It is interesting to note that the years since the enunciation of the agile philosophy of development have spawned more ideas and methodological practices than ever before. The incipient stages saw a proliferation of articles on the challenges of moving to agile (Boehm and Turner 2005; Nerur et al. 2005), the co-existence of traditional and agile methods (e.g., ambidexterity) (Ramesh et al. 2012; Vinekar et al. 2006), the efficacy of pairs vis-à-vis individuals (Balijepally et al. 2009; Cockburn and Williams 2001), the conceptual underpinnings of agile development, and ways to precisely define and measure agility (Nerur and Balijepally 2007; Nerur et al. 2010), to name but a few. Questions about its efficacy in distributed/virtual environments as well as in outsourced/offshored projects surfaced as well (Sarker and Sarker 2009). As other practices, such as refactoring and test-driven development, began to gain currency, the conversation shifted to issues (e.g., benefits, change in mindsets) related to these practices (Crispin 2006; Janzen and Saiedian 2008; Shull et al. 2010). Ideas on the role of lean thinking and lean governance on agile practices soon followed (Greer and Hamon 2011; Poppendieck and Poppendieck 2006). While this evolution of thoughts and ideas in the agile domain may be apparent to a few experts, the exact nature of the shifts in thinking is yet to be examined in the literature. The primary aim of our paper is to discern the changes in concepts related to agile over the last fifteen years. Since words are the means through which scholars express their ideas, an analysis of the distribution of words across a corpus

has the potential to reveal the underlying intellectual structure of the domain of interest. Based on this premise, we employ text analytics to construct topics pertinent to two consecutive time periods that have witnessed the maturation of concepts in agile development.

The remainder of the paper is organized along the following lines. The next section discusses prior literature, followed by a description of the methodology employed in this study. Subsequently, we present our results and discuss their implications. Limitations of the study are then reviewed before concluding with a synopsis of the paper and directions for future research.

## Understanding the Intellectual Structure of a Field

Articles—whether they are published in scholarly outlets or in trade journals and magazines—are repositories of knowledge, encapsulating ideas and concepts that often engender new approaches and best practices in every field of human endeavor. Over the years, some ideas endure as they mature and spawn new ones, while others may just fade away. Understanding this conceptual structure is important for several reasons. First, it gives us some sense of where we began our intellectual journey and where we are today. Second, it reminds us that not all concepts persist. More importantly, it helps us to reflect on the reasons why some ideas endure and others don't. Third, it has the potential to reveal the trajectory of the field and to anticipate what is forthcoming. Since our study is concerned with precisely this endeavor, we review various quantitative approaches that have been used in the past to delineate the ideational structure of fields.

Scholars have primarily relied on bibliometric techniques such as Author Co-Citation Analysis (ACA) and Document Co-Citation Analysis (DCA) to examine distinct thematic areas within their disciplines. For example, these techniques have been used to unravel the research streams underlying several disciplines, including strategic management (Nerur et al. 2008; Ramos-Rodríguez and Ruíz-Navarro 2004), management information systems (Culnan 1986; Culnan 1987), international management (Acedo and Casillas 2005), and information science (White and McCain 1998), to name but a few research fields. In fact, an earlier effort at constructing the conceptual structure of agile software development was made by Torgeir, Nerur, Balijepally, and Moe (2012), who used ACA to show the main research themes within the field.

While authors constitute the units of analysis in ACA, documents are the objects of interest in DCA. The salient features as well as shortcomings of these approaches are provided below.

1. In both ACA and DCA, the unit of analysis – author or document – is really a placeholder for the idea(s) that it exemplifies (Culnan 1987).
2. Frequency of co-citations between pairs of authors/documents provides a similarity matrix that can be processed to derive clusters, factors, and a multidimensional scaling map. The underlying assumption is that authors (or documents) that are frequently cited together embody knowledge/ideas related to a specific subfield or research stream within a discipline.
3. The latent semantic relationships between the writings of authors or documents are not directly examined by these approaches. Instead, the similarity structure that unfolds is based largely on the perceptions of citing authors.
4. ACA and DCA do not factor in the context in which the citation occurs. For example, no distinction is made between citations that support or build on an article versus those that don't.
5. Yet another limitation of ACA and DCA is that they have relied largely on seminal or most cited authors/documents for elucidating the structure of the field. This not only excludes authors and documents that have not garnered enough citations, but also ignores good ideas that may be latent in articles that for some reason have not been cited frequently.

In addition to ACA and DCA, one may also use social network analysis to assess the status of a field. For example Acedo, Barroso, Casanueva and Galán (2006) used co-authorship among researchers in the organizational and management field to construct a social network. The network was then subjected to social network analysis (SNA).

As mentioned earlier, neither ACA nor DCA uses the semantics of the documents/writings of authors to evolve a conceptual structure. The goal of our study is not to just delineate the intellectual streams within agile, but also to understand how the vocabulary of the field has changed over the two time periods of interest. Specifically, our paper endeavors to understand the key topic areas and their associated words for each time period. Hence, our paper employs text analytics to gain insights into the semantic structure of the field of agile software development.

The next section provides an overview of text analytics and describes the steps that were used in this study.

## Overview of Text Analytics & Text Mining

The abundance of unstructured data has brought text analytics and text mining to the forefront of research. The proliferation of unstructured text in social media outlets (e.g., Twitter, Linkedin, Facebook) as well as in blogs and formal publication outlets (e.g., journals and magazines) provides ample opportunities for researchers to examine latent semantic relationships using traditional data mining techniques (Weiss et al. 2010). These techniques, however, require numerical data; therefore, we transform raw text into vectors of numbers that can be subjected to quantitative analysis (Miller 2015).

The steps involved in downloading the text, preprocessing it, and eventually converting it to numerical vectors are arguably more cumbersome than the analysis itself. The various formats in which text may be downloaded present a challenge. Text data on the Internet comes in a variety of formats, including JSON (Javascript Object Notation), XML (Extensible Markup Language), PDF (Portable Document Format), and csv/tsv (comma or tab separated values). Fortunately, contemporary tools enable researchers to retrieve relevant text from these formats fairly easily.

Once the raw text is obtained, it needs to be cleaned up before it can be used for analysis. To facilitate word frequency counts as well as matches across documents, it is customary to convert the text to lowercase. This is followed by the removal of commonly occurring English words such as “of” and “the” (these are referred to as “stopwords”) as well as elimination of punctuation and digits. A list of user-selected stopwords from the domain being investigated may also have to be dropped from the text. Subsequently, the researcher may employ a technique called “stemming” to reduce some words to their root form, so that they are treated alike during analysis (Miller 2015; Weiss et al. 2010). For example, “method” and methods” really refer to the same concept, as do words such as “agile” and “agility” or “develop”, “developing” and “development”. Reduction of these words to their “stems” will enable us to treat all these words as being equivalent.

The next step is to generate a vector of numbers based on word frequency counts or a score/weight that indicates the importance of the word. A formulation referred to as Term Frequency – Inverse Document Frequency (tf-idf) is often used for the latter (see (Weiss et al. 2010)). In this formulation, rare words are accorded more importance. Likewise, terms that occur frequently within a document get higher scores than those that appear in all the documents. The interested reader may refer to Manning, Raghavan, and Schütze (2008) for further details.

The vector of numbers obtained using a tf-idf vectorizer may be used to derive a Term-Document Matrix (or a Document Term Matrix). As the name implies, the Term- Document Matrix (TDM) would have terms/words as rows and documents as columns. Such a matrix can then be used to construct a distance (e.g., cosine similarity) matrix to assess similarities of documents, derive clusters, and so forth. TDM may also be used as an input to Non-Negative Matrix Factorization (NNMF), a technique that is increasingly finding favor in applications that require decomposition/dimensionality reduction of large multivariate data (Hoyer 2004; Lee and Seung 1999). The fact that our TDM contains non-negative data fulfills the condition for using NNMF.

The NNMF algorithm requires the user to specify the number of topics desired (say, f). The technique factorizes the original TDM of dimensions t x d (t = number of terms and d = number of documents) into two smaller matrices, say A and B, having dimensions t x f and f x d, respectively. All three matrices will

have non-negative cell values. Specifically, the value in the cell $TDM_{ij}$ is a count of the number of times the $i^{th}$ word occurs in the $j^{th}$ document. The f topics requested constitute semantic features that are represented by the columns in A. An iterative procedure then attempts to approximate the factorized matrix using A and B [for example, see (Hoyer 2004; Lee and Seung 1999)]. The outcome is the desired number of topics and their associated words.

The preceding discussions are summarized in Figure 1. The next section presents our results and discusses the implications of our research.

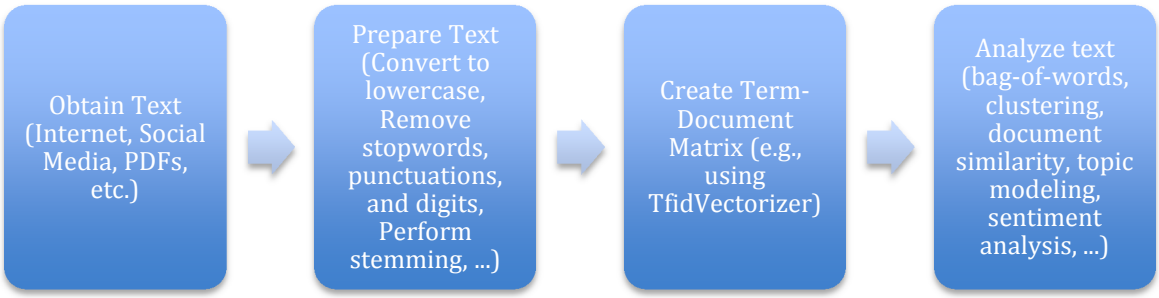


**Figure 1. Steps in Text Analytics**
Source: Adapted from Miller(2015)

## Results & Discussion

### *Data Collection*

As mentioned earlier, the primary objective of our paper is to assess the intellectual structure of Agile Software Development. Specifically, we are interested in the evolution of ideas and shifts in thinking that have occurred in the last fourteen years, starting from 2001, the year agile was conceived, to 2014. To get a better sense of how different the research themes in the early years were compared with topics of interest today, two time periods were analyzed. The first time period from 2001 to 2007 represents the formative years of the field, while the period from 2008 to 2014 captures themes that have dominated the conversation in recent times.

Since one of our goals was to uncover the latent semantic structure of the field, we used text analytics in our study. The steps outlined below give a detailed account of our data collection efforts.

1. For each of the two time periods, abstracts of all articles that satisfied our search criteria[1] were downloaded from the Web of Science database. There were 796 downloadable abstracts for the first time period and 1475 for the second.

[1] Search criteria used were:
**TOPIC:** ("agile development") *OR* **TOPIC:** ("agile software development") *OR* **TOPIC:** ("agile methodologies") *OR* **TOPIC:** ("agile methods") *OR* **TOPIC:** ("agile project management") *OR* **TOPIC:** ("lean development") *OR* **TOPIC:** ("lean software development") *OR* **TOPIC:** ("scrum") *OR* **TOPIC:** ("extreme programming") *OR* **TOPIC:** ("pair programming") *OR* **TOPIC:** ("test-driven development") *OR* **TOPIC:** ("test driven development") *OR* **TOPIC:** ("distributed agile") *OR* **TOPIC:** ("global agile") *OR* **TOPIC:** ("large-scale agile") *OR* **TOPIC:** ("large scale agile")
**Refined by: RESEARCH AREAS:** ( COMPUTER SCIENCE OR ENGINEERING OR BUSINESS ECONOMICS OR TELECOMMUNICATIONS OR INFORMATION SCIENCE LIBRARY SCIENCE OR OPERATIONS RESEARCH MANAGEMENT SCIENCE )

2. The corpus for 2001-2007 thus contained 796 abstracts stored in separate files. Likewise, the corpus used to derive the latent semantic structure of the period 2008-2014 comprised 1475 files, each containing the abstract of an article that satisfied our search criteria.
3. Following standard practices in text analytics (see preceding section), the corpus of each period was analyzed. Although stemming could have eliminated some duplication of words, particularly plurals (e.g., "project" and "projects" were treated as distinct words), the stem words that resulted from a preliminary analysis were not very helpful in understanding the semantics. Hence, stemming was used only for the word frequency plots. The results and their implications are discussed in the next sub-section.

### *Results*

Figures 2 and 3 show the frequency plots of the top 25 words in each of the time periods. Note that the words are stemmed and therefore what we see are the "stem roots", which, at times, can be difficult to interpret. It is apparent from the two graphs that there is a great deal of overlap between the words frequently employed in the first time period and those that appear often in the second. However, the words "XP", "Extreme" (stemmed to "extrem"), and "pair" that appeared quite frequently in the first time period are absent from the 25 most often used words in the period 2008-2014. Further, the word "Scrum" shows up in the second time period but not in the first, suggesting that Scrum has replaced XP as the most dominant agile method. The figure also suggests that Scrum and "projects" concerns are more evident in the latter years.

Much of the early research on agile was on XP, pairs versus individuals, challenges of transitioning to agile, issues related to process, requirements, team performance, and quality of outcomes. As Scrum gained in popularity, concerns about its efficacy in agile projects began to dominate the conversation. It also appears that issues related to testing and test-driven development have been quite popular in both time periods. Distributed agile development and lean concepts/principles are conspicuously absent.

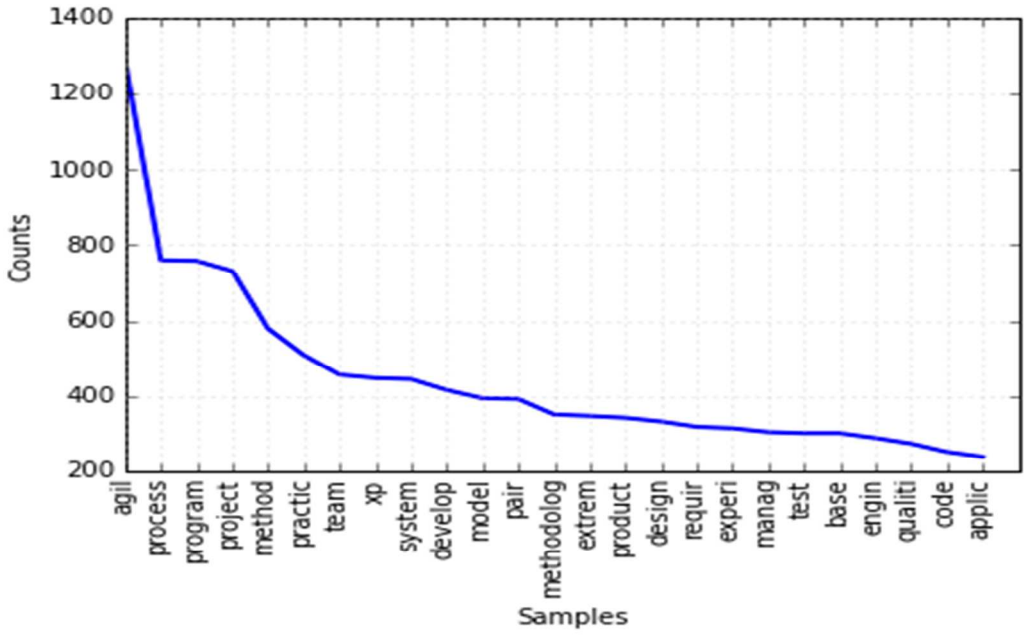


**Figure 2. Frequency Plots of Top 25 Words (2001-2007)**

---

**Timespan:** 2001-2007.
Search language=Auto

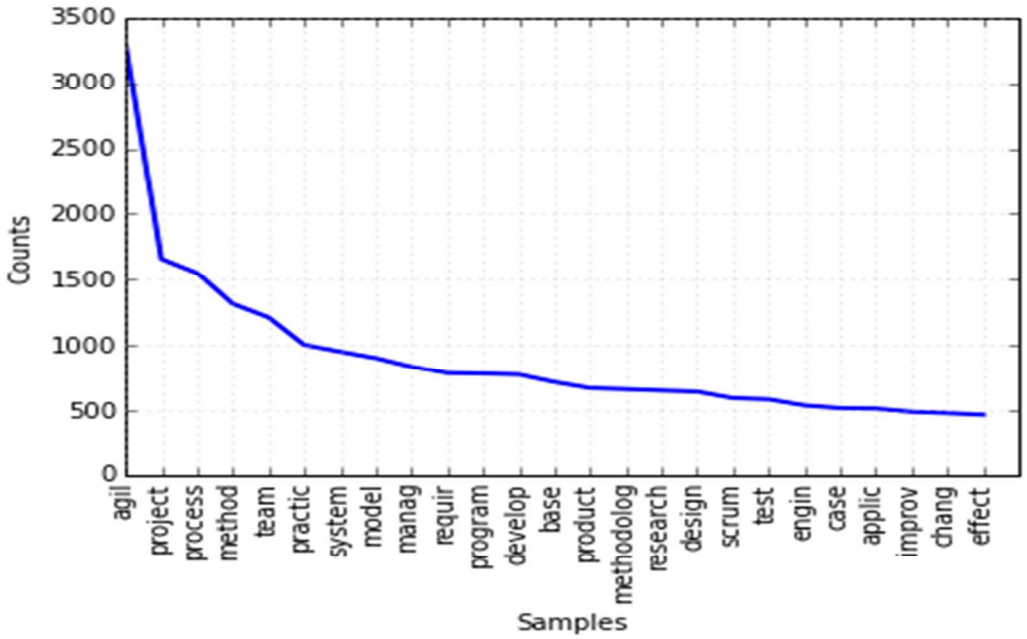


**Figure 3. Frequency Plots of Top 25 Words (2008-2014)**

While many of the words in Figures 2 and 3 are the same, there are some conspicuous differences as well. Figure 2 includes words such as "XP", "pair", "extreme", and "code" that show the prominence of articles related to eXtreme Programming, an agile method that was dominant in the early years agile. The word "Scrum" figures among the top 25 words in the second period but not in the first. These differences suggest a shift in research from XP to Scrum in more recent times. Since word frequencies alone are inadequate to draw any insights, a topic analysis was performed to understand these shifts better. A topic may be regarded as an intellectual stream or research theme. A technique called Non-Negative Matrix Factorization (NNMF) was performed on key features in the abstracts to elicit the topics.

The results of NNMF for the two time periods are shown in Table 1. As mentioned earlier, NNMF is a data reduction technique that is eminently suitable for identifying topics from a corpus of text. The tables show the topics, the key words or concepts associated with the topic, and the label that we have provided for the topic area.

While the overlap in research themes is evident from Table 1, this analysis does reveal areas of research that have become prominent in the agile software development field. The shift in thinking to distributed agile, lean thinking, and large scale systems is also apparent in Table 1. It is rather surprising that User Experience Design (UXD), arguably one of the most important areas for today's deployment platforms such as smartphones and tablets, does not figure prominently in the second time period.

While our results help us understand what has been done so far and what the dominant contemporary themes are, it also forces us to examine the gaps in the literature and reflect on areas that should have gained momentum. There is an imperative need to do more research related to governance (lean and other forms), portfolio approach to agile project management, scalability of agile projects, risk management in agile, integration of lean thinking with agile practices, and evolving measures for concepts such as "agility", "self-organization" and "complexity", to name but a few.

| Topic | Top words associated with topic (2001-2007) | Topic Label (2001-2007) | Top words associated with topic (2008-2014) | Topic Label (2008-2014) |
|---|---|---|---|---|
| 1 | agile methods practices methodologies projects organizations method traditional research large systems experiences agility principles organization processes change gaining developing developers | Agile vs Traditional | agile methods practices research methodologies adoption method projects agility traditional principles literature survey processes practitioners case analysis organizations organizational engineering | Agile vs Traditional |
| 2 | pair programming programmers pairs experiment distributed solo empirical benefits tasks productivity quality programmer single experiments developers computer code phase cost | Pair Programming | programming pair students learning computer student pairs pp solo courses programmers teaching introductory skills personality experiment program developers collaborative education | Pair Programming, Collaboration, and Learning |
| 3 | testing code web applications tests application driven systems framework automated tool source tools based refactoring cases service specification language security | Automated Testing | team teams distributed communication members agile collaboration performance challenges meetings effective self coordination theory collocated tools time developers customer factors | Coordination and communication in distributed agile teams |
| 4 | xp extreme practices programming methodology practice customer research values principles known environment developers good perspective reports method companies best small | Extreme Programming | project management projects risk knowledge success managers product traditional planning value implementation model agile based schedule control successful plan customer | Agile Project Management |
| 5 | students learning student engineering computer teaching experience courses programming undergraduate collaborative education academic experiences curriculum carolina introductory classes assignments program | Collaborative Learning in the classroom | tdd code driven quality practice source tests writing cases refactoring productivity developers coverage dice tool implementation case written developer benefits | Test-Driven Development |
| 6 | tdd driven code tests practice write quality writing developers productivity programmers automated programmer developer experiment empirical experiments effectiveness cases written | Test-Driven Development | lean product business value innovation principles line large products waste organization customer improvement companies thinking processes process time industry scale | Lean Thinking |
| 7 | team teams members knowledge distributed agile communication experience face practices organization self multiple time individual changes productivity suggests project build | Knowledge and communication in distributed agile teams | xp extreme process practices programming method methodology customer release quality cmmi values improvement model planning medium team case models secure | XP vs CMMI |
| 8 | process model processes engineering models requirements based improvement method extreme management products knowledge systems concepts cmm programming maturity methodology level | Process Models & Requirements Engineering | requirements process design systems model user based engineering web application framework models applications users methodologies processes business architecture modeling methodology | Requirements Engineering & Modeling |
| 9 | design user quality interaction usability designing evolution products code designs architectural interface phase database product iterative ui question evaluation understand | User Experience Design (UXD) | testing tests acceptance usability driven customer user automated cases tool techniques executable gui tools integration language regression web behavior release | Automated Testing and Integration |
| 10 | project management customer projects product value business agile managers communication time scrum environment large company successful requirements quality customers planning | Scrum Project Management | scrum process projects cmmi sprint experience small case company teams methodology students framework improve open model companies method productivity decision | Scrum vs CMMI |

**Table 1. Key Topics of research in the two time period**

To get a sense of how our approach compares with traditional bibliometric techniques such as ACA, we compared our results with those of the study by Torgeir et al. (2012, pp. 1217). The ACA approach identified research areas such as design patterns, distributed cognition, software estimation, and the prevalence of case-study methodology, which were not apparent in our topics. However, our study revealed several themes that were not evident in the paper by Torgeir et al. These include: Automated Testing, Collaborative Learning, Test-Driven Development, Knowledge/Coordination/Communication in distributed agile, Lean Thinking, XP vs CMMI, and Scrum Project Management. There is, therefore, a strong indication that topic analysis can complement and extend the scope of traditional bibliometric techniques.

## Limitations

Our study, not unlike other research endeavors, has some limitations. First, data that we deal with in text analytics are inherently noisy. A combination of Unix scripts and Python programs were used to parse the text (e.g., removing punctuations and numbers, converting to lowercase, removing common English stopwords as well as user-identified terms). For example, our search criteria included an abstract or two that contained the term "bovine scrum albumin", which had nothing to do with agile software development. Given that an overwhelming majority of the abstracts were relevant to our study, it is very unlikely that these spurious abstracts would have altered our results in any major way. Second, our study used only abstracts, which are poor substitutes for a corpus of complete papers. A more accurate and comprehensive view of the semantic structure may be revealed by an analysis of all the papers rather than their abstracts. Having said that, we believe our effort is a good first approximation of the lexical structure of agile software development. Third, the search criteria used determines the abstracts that are retrieved. It is conceivable that the addition of a few search terms (e.g. "scrumban") might have elicited some more abstracts pertinent to our study. Again, given the volumes of relevant abstracts that were examined, we are reasonably confident that these shortcomings do not detract from the findings in any significant way. Lastly, the choice of 10 topics set for data reduction through Non-Negative Matrix Factorization (NNMF) was somewhat arbitrary. However, most studies on ACA/DCA reveal no more than 10 subfields, so our choice of 10 topics is not altogether limiting.

## Conclusions and Future Directions

The abundance of unstructured textual data presents both a challenge and an opportunity to researchers. The computational techniques that fall under the umbrella of "text analytics" have the potential to offer insights that may not be forthcoming with traditional approaches. In our exploratory study, we use text analytics to delineate the scholarly landscape of agile software development. We argue that our approach, in contrast to prior bibliometric techniques, can explore the latent semantic relationships present in a corpus to offer a richer interpretation of the intellectual structure of a field. Our findings suggest that many of the research areas that were of primary concern in the first seven years persist in the subsequent period of analysis. It also shows that research related to distributed agile, lean thinking and large-scale systems, and Scrum & Agile Project Management have occupied researchers in recent times.

Our study makes several contributions. First, it shows the potential of text analytics in explicating hidden semantic relationships in a large body of text. Second, it demonstrates the utility of text analytics in complementing erstwhile techniques (e.g., bibliometrics) that have been employed to understand the intellectual structure of fields. Third, the topic modeling technique used here may be used to understand other sources of unstructured text, such as social media data. Fourth, it reveals the conceptual structure of agile software development and its key research streams

Future research may extend this study by: a) Examining complete papers rather than just abstracts; b) Comparing the performance of NNMF with other topic modeling approaches (e.g., Latent Dirichlet Allocation); and c) Combining text analytics with other methods to provide multiple perspectives on the conceptual structure of a domain.

## References


Acedo, F.J., Barroso, C., Casanueva, C., and Galán, J.L. 2006. "Co-Authorship in Management and Organizational Studies: An Empirical and Network Analysis*," *Journal of Management Studies* (43:5), pp 957-983.

Acedo, F.J., and Casillas, J.C. 2005. "Current Paradigms in the International Management Field: An Author Co-Citation Analysis," *International Business Review* (14:5), pp 619-639.

AgileAlliance. 2001. "Manifesto for Agile Software Development." Retrieved Jun 1, 2013, from http://www.agilemanifesto.org

Ambler, S.W. 2013. "How Agile Are You? 2013 Survey." Retrieved Sep 5, 2013, from http://www.ambysoft.com/surveys/howAgileAreYou2013.html

Balijepally, V., Mahapatra, R., Nerur, S., and Price, K.H. 2009. "Are Two Heads Better Than One for Software Development? The Productivity Paradox of Pair Programming," *MIS Quarterly* (33:1), pp 91-118.

Boehm, B. 2002. "Get Ready for Agile Methods with Care," *IEEE Computer* (35:1), pp 64-69.

Boehm, B., and Turner, R. 2005. "Management Challenges to Implementing Agile Processes in Traditional Development Organizations," *IEEE Software* (22:5), pp 30-39.

Cockburn, A., and Highsmith, J. 2001. "Agile Software Development 2: The People Factor," *IEEE Computer* (34:11), pp 131-133.

Cockburn, A., and Williams, L. 2001. "The Costs and Benefits of Pair Programming," in: *Extreme Programming Examined,* G. Succi and M. Marchesi (eds.). Boston, MA: Addison Wesley, pp. 223-243.

Crispin, L. 2006. "Driving Software Quality: How Test-Driven Development Impacts Software Quality," *IEEE Software* (23:6), pp 70-71.

Culnan, M.J. 1986. "The Intellectual Development of Management Information Systems, 1972-1982: A Co-Citation Analysis," *Management Science* (32:2), p 156.

Culnan, M.J. 1987. "Mapping the Intellectual Structure of Mis, 1980-1985: A Co-Citation Analysis.," *MIS Quarterly* (11:3), p 340.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N.B. 2012. "A Decade of Agile Methodologies: Towards Explaining Agile Software Development," *Journal of Systems & Software* (85:6), pp 1213-1221.

Greer, D., and Hamon, Y. 2011. "Agile Software Development," *Software: Practice and Experience* (41:9), pp 943-944.

Highsmith, J., and Cockburn, A. 2001. "Agile Software Development 1: The Business of Innovation," *IEEE Computer* (34:9), pp 120-127.

Hoyer, P.O. 2004. "Non-Negative Matrix Factorization with Sparseness Constraints," *J. Mach. Learn. Res.* (5), pp 1457-1469.

Janzen, D.S., and Saiedian, H. 2008. "Does Test-Driven Development Really Improve Software Design Quality?," *IEEE Software* (25:2), pp 77-84.

Lee, D.D., and Seung, H.S. 1999. "Learning the Parts of Objects by Non-Negative Matrix Factorization," *Nature* (401:6755), pp 788-791.

Manning, C.D., Raghavan, P., and Schütze, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Miller, T.W. 2015. *Modeling Techniques in Predictive Analytics with Python and R: A Guide to Data Science*. Upper Saddle River, NJ: Pearson Education.

Nerur, S., and Balijepally, V. 2007. "Theoretical Reflections on Agile Development Methodologies," *Communications of the ACM* (50:3), pp 79-83.

Nerur, S., Cannon, A., Balijepally, V., and Bond, P. 2010. "Towards an Understanding of the Conceptual Underpinnings of Agile Development Methodologies," in: *Agile Software Development: Current Research and Future Directions,* T. Dingsøyr, T. Dybå and N.B. Moe (eds.). Springer, pp. 15-29.

Nerur, S., Mahapatra, R., and Mangalaraj, G. 2005. "Challenges of Migrating to Agile Methodologies," *Communications of the ACM* (48:5), pp 73-78.

Nerur, S.P., Rasheed, A.A., and Natarajan, V. 2008. "The Intellectual Structure of the Strategic Management Field: An Author Co-Citation Analysis," *Strategic Management Journal* (29:3), pp 319-336.

Poppendieck, M., and Poppendieck, T. 2006. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley Professional

Ramesh, B., Mohan, K., and Cao, L. 2012. "Ambidexterity in Agile Distributed Development: An Empirical Investigation," *Information Systems Research* (23:2), pp 323-339.

Ramos-Rodríguez, A.-R., and Ruíz-Navarro, J. 2004. "Changes in the Intellectual Structure of Strategic Management Research: A Bibliometric Study of the Strategic Management Journal, 1980-2000," *Strategic Management Journal* (25:10), pp 981-1004.

Sarker, S., and Sarker, S. 2009. "Exploring Agility in Distributed Information Systems Development Teams: An Interpretive Study in an Offshoring Context," *Information Systems Research* (20:3), pp 440-461.

Shull, F., Melnik, G., Turhan, B., Layman, L., Diep, M., and Erdogmus, H. 2010. "What Do We Know About Test-Driven Development?," *IEEE Software* (27:6), pp 16-19.

Torgeir, D., Nerur, S., Balijepally, V., and Moe, N.B. 2012. "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software*, 85, pp. 1213-1221.

Vinekar, V., Slinkman, C.W., and Nerur, S. 2006. "Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View," *Information Systems Management* (23:3), pp 31-42.

Weiss, S.M., Indurkhya, N., and Zhang, T. 2010. *Fundamentals of Predictive Text Mining*. Springer London.

White, H.D., and McCain, K.W. 1998. "Visualizing a Discipline: An Author Co-Citation Analysis of Information Science, 1972–1995," *Journal of the American Society for Information Science* (49:4), pp 327-355.