Extracción de Nubes de Palabras en Repositorios Git

Extracting Word Clouds in Git Repositories

Paul Mendoza del Carpio
Departamento de Ingeniería de Software
Universidad La Salle
Arequipa, Perú
p.mendozadc@ulasalle.edu.pe

Resumen — **Las plataformas de codificación social basadas en Git (e.g.; GitHub, BitBucket) han sido adoptadas por una importante cantidad de proyectos de código abierto, y se han convertido en una importante fuente de información técnica y social acerca del desarrollo de software. Tal información podría ser empleada para identificar convenciones de programación utilizadas por equipos de desarrollo de software. El presente trabajo pretende explotar los patrones de nombrado inmersos en un repositorio Git de código fuente, para obtener nubes de palabras como resumen de éste. Repositorios Git de diferentes proyectos de código abierto han sido empleados para evaluar la propuesta. Los resultados muestran palabras significativas extraídas de los repositorios, y características frecuentes de implementación para artefactos de código que utilizan tales palabras.**

Palabras Clave – patrón de nombrado; nubes de palabras; repositorio Git.

Abstract — **Git based social coding platforms (e.g.; GitHub, BitBucket) have been broadly adopted by many open source projects, and have became an important source of technical and social information about software development. That information could be employed for identifying programming conventions used by software development teams. This work intends leverage the naming patterns immersed in a Git repository of source code, for getting word clouds as a summary of it. Git repositories of several open source projects have been employed to evaluate the proposal. The results show significant words mined from the repositories, and frequent implementation features for code artifacts which use these words.**

Keywords – naming pattern; word cloud; Git repository.

I. INTRODUCCIÓN

Conforme un sistema de software evoluciona, su código fuente se convierte en una fuente de información que se encuentra actualizada y que contiene información relevante acerca del dominio de la aplicación [18]. La relevancia de los nombres de artefactos de código (e.g.; clase) radica en su aporte al código fuente para ser legible, mantenible y relacionado al dominio del problema [28]. Se puede mencionar que los programadores en un equipo de desarrollo son conscientes de la importancia de los nombres establecidos en sus unidades de código [10], por ello emplean nombres identificadores que puedan comunicar la función que cumplen [26], y procuran elegir nombres que sean convencionales (i.e., nombres usados en la industria y/o en comunidades de desarrollo) [27].

Asimismo, las plataformas de codificación social basadas en Git (e.g.; GitHub, BitBucket) ofrecen grandes ventajas para el desarrollo colaborativo [43]. Tales plataformas han sido adoptadas por una importante cantidad de proyectos de código abierto [4], y se han convertido en una importante fuente de información técnica y social acerca del desarrollo de software [3] [5]. Luego, tal información podría ser utilizada para identificar convenciones de programación utilizadas por equipos de desarrollo de software.

Por otro lado, las nubes de palabras son ampliamente adoptadas para resumir visualmente grandes cantidades de texto [9] [15] [17] [22] [25] [42], éstas presentan el contenido en una forma concisa, con el tamaño de fuente mapeado a la frecuencia o importancia de las palabras presentes [15] [22] [24], permitiendo obtener rápidamente una comprensión de la data subyacente [36].

El presente trabajo bosqueja un método que pretende explotar las convenciones de nombrado inmersas en un repositorio Git de código fuente, para obtener un resumen de éste en forma de palabras significativas visualizadas mediante nubes de palabras. Este trabajo emplea los nombres de artefactos de código para extraer palabras significativas de un sistema de software y mostrar información de resumen mediante nubes de palabras, siendo aplicable sobre repositorios Git que constituyen una fuente rica de convenciones y prácticas. La principal contribución del presente trabajo es el de proporcionar: (1) un método para extraer palabras significativas desde repositorios Git de código fuente, (2) una forma de uso de las nubes de palabras para mostrar palabras significativas y sus características para artefactos de una aplicación de software, las cuales pueden representar implementaciones bien definidas apoyadas por su frecuencia y cantidad.

II. TRABAJOS RELACIONADOS

Las nubes de palabras han sido empleadas para el resumen de texto y análisis en diversos dominios [15]. A fin de identificar trabajos relacionados al presente, se han realizado búsquedas sobre las bibliotecas digitales ACM, IEEE Xplore y ScienceDirect. La Tabla I muestra las consultas realizadas y la cantidad de resultados (CR) obtenidos para cada biblioteca.

Respecto a los resultados en ACM, las nubes de palabras son utilizadas con los siguientes propósitos: mostrar identificadores presentes dentro del código para identificar la distribución de clones de código [21], mostrar los términos más

frecuentes en el código y mejorar las capacidades de búsqueda [31], mostrar palabras populares para facilitar la formulación de consultas de búsqueda [40], mostrar resumen visual de data y metadata de servicios [39]; otros resultados no están relacionados a código fuente. Respecto a los dos resultados en IEEE Xplore, [31] coincide con el obtenido en ACM, y el otro resultado sólo menciona la generación de nubes de palabras sobre texto de una lista de correos. En ScienceDirect no se encontraron trabajos que empleen nubes de palabras relacionadas directamente al código fuente. Según los resultados obtenidos, no se habrían encontrado trabajos acerca de nubes de palabras, enfocados al código fuente en una forma similar al presente trabajo.

TABLA I. CONSULTA DE TRABAJOS RELACIONADOS

Biblioteca	*Consulta*	*CR*
ACM	("code" OR "coding") AND recordAbstract:("word cloud" "word clouds" "tag cloud" "tag clouds")	10
IEEE Xplore	("code" OR "coding") AND ("Abstract":"word cloud" OR "Abstract":"word clouds" OR "Abstract":"tag cloud" OR "Abstract":"tag clouds")	2
ScienceDirect	("code" OR "coding") AND (ABS("word cloud") OR ABS("word clouds") OR ABS("tag cloud") OR ABS("tag clouds"))	19

III. PATRONES DE NOMBRADO

El uso de convenciones de nombrado es predominante en el desarrollo de software [28]. En este trabajo se pretende explotar tal uso con el fin de identificar palabras frecuentes que son utilizadas en nombres de artefactos de código dentro de un repositorio Git. Tales palabras son identificadas por contar con una frecuencia mínima de ocurrencia al cumplir con los patrones de nombrado presentados a continuación; se muestran ejemplos reales encontrados en repositorios Git de las fundaciones Apache y Eclipse.

A. Patron 1: Nombre y Nivel de Paquete

La palabra es empleada frecuentemente en clases incluidas en paquetes con un mismo nombre y a un mismo nivel de la jerarquía de paquetes. La figura 1 muestra paquetes de Apache Hadoop que siguen este patrón. Se define a f como un valor de frecuencia mínima; N es el conjunto de niveles de paquetes; G(n,m) es el conjunto de paquetes con nombre m que se encuentran ubicados en el nivel n ∈ N; y G(n,m,p) es el conjunto de paquetes de nivel n con nombre m que contienen clases que usan la palabra p. Las palabras p de este patrón son tales que (| G(n,m,p) | / | G(n,m) |) ≥ f, y | G(n,m) | > 2.

B. Patrón 2: Nombre de Paquete

La palabra es empleada frecuentemente en clases incluidas en paquetes con un mismo nombre. La figura 2 muestra paquetes de Eclipse EGit que siguen este patrón. Se define a M como el conjunto de nombres de paquetes; F(m) es el conjunto de paquetes con nombre m ∈ M; y F(m,p) es el conjunto de paquetes con nombre m que contienen clases que usan la palabra p. Las palabras p de este patrón son tales que (| F(m,p) | / | F(m) |) ≥ f, y | F(m) | > 2.

C. Patrón 3: Paquete Superior Inmediato

La palabra es empleada frecuentemente en clases incluidas en paquetes que pertenecen a un mismo paquete superior. La figura 3 muestra paquetes de Eclipse BPMN2 que siguen este patrón. Se define a H(q) como el conjunto de paquetes contenidos en el paquete q; y H(q,p) es el conjunto de paquetes contenidos en q que contienen clases que usan el término p. Las palabras p de este patrón son tales que (| H(q,p) | / | H(q) |) ≥ f, y | H(q) | > 2.

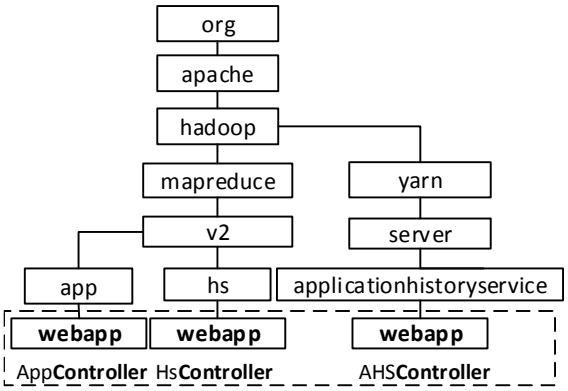

Figura 1. Ejemplo del patrón Nombre y Nivel de Paquete

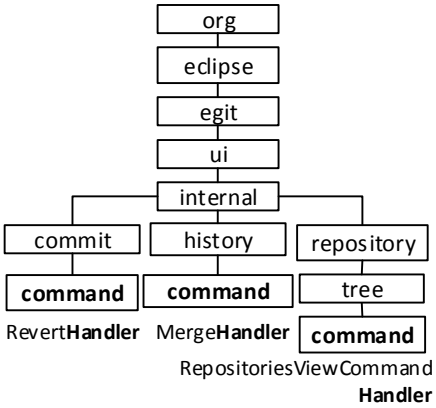

Figura 2. Ejemplo del patrón Nombre de Paquete

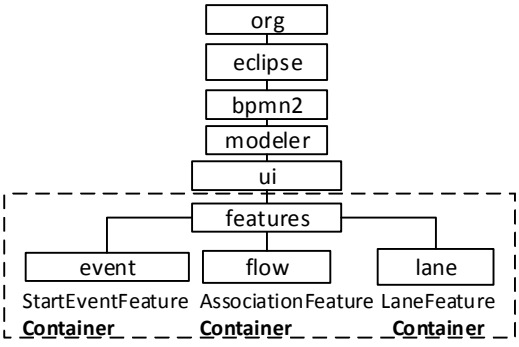

Figura 3. Ejemplo del patrón Paquete Superior Inmediato

D. Patrón 4: Paquete

La palabra es empleada frecuentemente en clases incluidas en un mismo paquete. La figura 4 muestra paquetes de Apache MyFaces que siguen este patrón. P es el conjunto de las palabras usadas en nombres de clases; Q es el conjunto de los paquetes; C(q) es el conjunto de las clases de q ∈ Q; y C(q,p) es el conjunto de las clases de q que usan la palabra p ∈ P. Las palabras p de este patrón son tales que (| C(q,p) | / | C(q) |) ≥ f, y | C(q) | > 2.

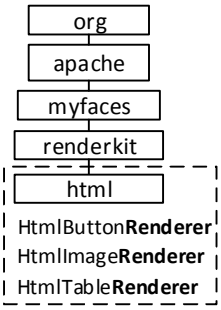

Figura 4. Ejemplo del patrón Paquete

IV. NUBES DE PALABRAS

Formalmente, se puede definir a una nube de palabras como un conjunto de palabras W = { w1, w2, . . . , wM } , donde cada palabra w ∈ W tiene atributos visuales que incluyen su tamaño de fuente, color y orientación (horizontal o vertical) [35]. Para seleccionar las palabras en una nube, se requiere una medición de la importancia de una palabra, denominada peso, la cual es típicamente la frecuencia del término (i.e.; tf, term frequency), pero también se tienen alternativas que emplean metadata [35]. Luego, el peso determina el tamaño de la fuente empleada para cada palabra [8] [37].

En este trabajo se propone una forma de uso de nubes de palabras sobre repositorios Git, mediante la presentación de dos tipos de nubes de palabras: la nube de palabras frecuentes (por repositorio Git), y la nube de características (para cada palabra de la nube de palabras frecuentes). Además, para ambos tipos de nubes se consideran dos tipos de presentaciones, de acuerdo a la medición del peso: por frecuencia y por cantidad (i.e.; cantidad de artefactos de código que cumplen el patrón de nombrado).

El método propuesto para extraer palabras significativas está conformado por dos etapas: (1) selección, y (2) definición de atributos. En la etapa (1), respecto a las nubes de palabras frecuentes, las palabras seleccionadas son aquellas que cumplen alguno de los patrones de nombrado presentados anteriormente, conforme a una frecuencia mínima establecida. Respecto a las nubes de características, ellas son generadas por cada palabra de la nube de palabras frecuentes, y muestran las características comunes encontradas en los artefactos de código fuente que emplean la palabra frecuente en el contexto del patrón de nombrado; tales características son seleccionadas para formar parte de la nube de acuerdo a una frecuencia mínima establecida. Se define a PF como el conjunto de las palabras frecuentes, AR(pf) es el conjunto de artefactos que emplean la palabra pf ∈ PF en el contexto de algún patrón de nombrado. Las nubes de características usan las siguientes frases para mostrar características frecuentes de unit ∈ AR(pf): extends S (S generaliza unit), implements I (unit realiza la interfaz I), T t1 (unit está asociado a T mediante un atributo de referencia), @A (unit contiene metadata de tipo A, e.g.; anotaciones de Java). En la etapa (2), se define principalmente el atributo de peso de cada palabra de acuerdo a los tipos de presentación mencionados anteriormente. Otros atributos pueden ser definidos de acuerdo a criterios del equipo de desarrollo, pero se brindan sugerencias para las nubes de palabras frecuentes. Para el color de la palabra, definir un color distinguido para una frecuencia absoluta (i.e.; 1, presente en todas las ocurrencias del patrón) y otros matices para frecuencias menores; ello permitiría detectar artefactos que no siguen la convención de nombrado. Para la orientación, mostrar la palabra en forma horizontal si la palabra sólo fue seleccionada por un patrón; en caso de ser seleccionada por más de un patrón, mostrarla en forma vertical, ello permitiría detectar una posible ambigüedad de la palabra. Adicionalmente, palabras reservadas (i.e.; stop words) pueden ser consideradas para palabras y características frecuentes.

V. EVALUACIÓN Y RESULTADOS

Para la evaluación de este trabajo se tomaron en consideración aplicaciones con repositorios alojados en la plataforma GitHub, véase la Tabla II que muestra datos del repositorio Git: Usuario (dueño del repositorio), Repositorio (nombre), CP (cantidad de palabras frecuentes), CC (cantidad de características). Se desarrolló una aplicación que recibe las URL de los repositorios Git, y recorre su contenido de paquetes y artefactos. Los nombres de los artefactos de código fueron divididos en palabras considerando el estilo CamelCase (estilo dominante por su facilidad de escritura y su adopción en lenguajes como Python, Java, C, C++ [6] [19]). Para obtener los nombres de paquetes, artefactos de código fuente y contenido de estos últimos se utilizó el API (i.e.; Application programming interface) de la plataforma Git.

Este trabajo usa palabras empleadas en nombres de artefactos de código, que en la mayoría de proyectos de actualidad corresponden a clases, tal vocabulario de palabras es aquel que contiene una mayor cantidad de palabras en común con otros vocabularios como el de nombres de atributos, funciones, parámetros y comentarios [38].

La figura 5 muestra la nube de palabras frecuentes por frecuencia para el repositorio Git de Pentaho Kettle, aquí las palabras de mayor tamaño habrían cumplido en mejor forma los patrones de nombrado. La figura 6 muestra la nube de palabras frecuentes por cantidad para el repositorio Apache Groovy, las palabras de mayor tamaño son aquellas que tienen mayor cantidad de artefactos donde se cumplió algún patrón de nombrado. Algunas palabras pueden aparecer en más de una ocasión en las nubes mencionadas, ello debido a que la palabra es frecuente siguiendo más de un patrón de nombrado. Ambas nubes podrían ayudar a identificar desde diferentes perspectivas: convenciones de nombrado, y posteriormente, convenciones de programación acerca de la implementación de artefactos que usen la palabra correspondiente. Las nubes de palabras frecuentes, por ser generadas a partir de nombres de artefactos de código, podrían ser de utilidad en aplicaciones

con una arquitectura de software bien definida, ello considerando que las palabras frecuentes de una aplicación habitualmente están asociados a conceptos del dominio de la aplicación [38], y el uso de convenciones de nombrado definidos por una arquitectura de software facilita actividades de comprensión de software [41]. En proyectos de software ágiles, donde las convenciones de nombrado no son siempre definidas formalmente [1], las nubes permitirían identificar palabras frecuentes que podrían formar una convención de nombrado.

TABLA II. REPOSITORIOS GIT EVALUADOS

Usuario	*Repositorio*	*CP*	*CC*
Apache	Camel	56	42
Apache	Groovy	28	15
Apache	Hadoop	49	30
Apache	OpenJPA	3	1
Apache	MyFaces	9	3
Apache	Spark	2	4
Apache	Tomcat	46	19
Apache	Wicket	61	7
Eclipse	EclipseLink	47	4
Eclipse	EMF Ecore	15	4
Eclipse	Egit	4	3
Eclipse	JDT	10	2
IntelliJ	IDEA UI	31	24
Jasper	Jasper Reports	180	10
Pentaho	Kettle	15	0
Pentaho	Mondrian	10	3
Pentaho	Reporting	90	59
RapidMiner	RapidMiner Studio	94	55
Red Hat	WildFly	15	5

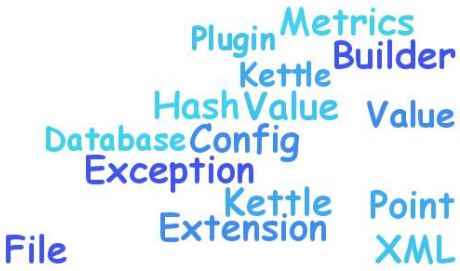

Figura 5. Nube de palabras frecuentes para Pentaho Kettle

La figura 7 muestra tres nubes de características, seguido se indica la palabra frecuente y sus características interpretadas: (1) Plugin de Pentaho Reporting es generalizada por AbstractActionPlugin, realiza la interfaz ControlActionPlugin y está asociada a ResourceBundleSupport, (2) Icon de RapidMiner Studio está asociada a Stroke y realiza las interfaces Icon, Serializable y UIResource, (3) Endpoint de Apache Camel presenta anotaciones de metadata UriPath, UriEndpoint y Metadata. Las nubes de características apoyarían la extracción de las estructuras emergentes en proyectos de software ágiles que siguen una arquitectura emergente (i.e.; nuevas estructuras emergen sin ser definidas en forma anticipada [30]). Asimismo, al extraer características comunes de varios artefactos de código que utilizan la misma palabra, se podrían identificar oportunidades de generalización y reutilización.

Figura 6. Nube de palabras frecuentes para Apache Groovy

Figura 7. Nubes de características

Las nubes de palabras son mecanismos adecuados para mostrar información de resumen debido a que son representaciones que proporcionan un resumen visual [7] [9] [34], y son estéticamente llamativas [11] [17] [29]. Asimismo, en el ámbito de repositorios de código Git, las nubes presentadas se pueden considerar mecanismos de resumen apropiados, debido a que son fáciles de interpretar [35] [42] (aún frente a un considerable volumen de repeticiones subyacente), evolucionan conforme a los cambios de las fuentes de datos a través del tiempo [29], son ubicuas en configuraciones en línea (i.e.; online) [9], y proporcionan una retroalimentación semántica para el análisis de un grupo de elementos que tienen algo en común [12]. Nótese que la generación de nubes de palabras frecuentes no requiere parsing del código fuente, sólo recorre el contenido de las rutas del repositorio Git, funcionalidad que habitualmente no tiene una tasa límite de uso en las API de plataformas Git. Adicionalmente, las nubes de palabras al ser aplicadas sobre

textos presentan algunas desventajas, como tratar variantes de una palabra como diferentes entradas [23], o no considerar conocimiento lingüístico [11], sin embargo en el contexto de repositorios de código ellas no serían relevantes.

Las palabras frecuentes extraídas del código fuente, pueden considerarse significativas puesto que las nubes de palabras muestran la significancia según la importancia (i.e.; peso) presentada en diferentes tamaños de fuente [2] [29]. Es válido considerar criterios adicionales para mostrar información significativa en nubes de palabras [2] como se ha utilizado en este trabajo.

Las nubes de palabras presentadas pueden servir como punto de inicio para análisis más profundos [11]. Es posible identificar tipos de componentes revisando nombres establecidos con una convención de nombrado [13] [20], y empleando los patrones de nombrado presentados [33]. Asimismo, este trabajo posibilitaría formar folksonomías para repositorios Git, proporcionando las palabras frecuentes como etiquetas, los artefactos de código como recursos, y los usuarios son tomados del repositorio; formando así el formal y bien aceptado modelo tripartito [14] para definir una folksonomía que emerge de varios individuos que han etiquetado contenido [16], en forma implícita al nombrar artefactos de código. En caso las nubes de palabras frecuentes se encuentren vacías, ello podría evidenciar prácticas de nombrado pobres, y/o deuda técnica arquitectónica [32].

El uso de nubes de palabras sobre el código fuente se puede diferenciar respecto a otros trabajos en: (1) obtiene palabras frecuentes en un nivel superior al de una unidad de código, (2) puede obtener conceptos relevantes a nivel de la aplicación (3) su resumen (nube de palabras frecuentes) al estar basado en nombres de artefactos, no requiere parsing de código fuente, siendo independiente del lenguaje de programación.

VI. CONCLUSIONES

Se ha presentado un método para extraer palabras significativas desde repositorios Git de código fuente, tales palabras son mostradas mediante dos presentaciones de nubes de palabras que resumen el contenido de un repositorio Git, basándose en patrones de nombrado que atienden la estructura jerárquica de paquetes (i.e.; directorios) del repositorio.

La forma de uso de las nubes de palabras ha servido para mostrar palabras significativas y características de artefactos en una aplicación de software. Estas últimas, apoyadas en su frecuencia y cantidad, han podido mostrar implementaciones comunes que apoyan la definición de convenciones de programación para artefactos de código que emplean la palabra significativa.

REFERÊNCIAS BIBLIOGRÁFICA

[1] Antonio Martini, Jan Bosch, and Michel Chaudron. 2015. Investigating Architectural Technical Debt accumulation and refactoring over time. *Inf. Softw. Technol.* 67, C (November 2015), 237-253. DOI=http://dx.doi.org/10.1016/j.infsof.2015.07.005

[2] Bjørnar Tessem, Solveig Bjørnestad, Weiqin Chen, and Lars Nyre. 2015. Word cloud visualisation of locative information. *J. Locat. Based Serv.* 9, 4 (October 2015), 254-272. DOI=http://dx.doi.org/10.1080/17489725.2015.1118566

[3] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G.J. van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and Tenure Diversity in GitHub Teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 3789-3798. DOI: https://doi.org/10.1145/2702123.2702549

[4] Christian Rodríguez-Bustos and Jairo Aponte. 2012. How distributed version control systems impact open source software projects. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories* (MSR '12). IEEE Press, Piscataway, NJ, USA, 36-39.

[5] Claudia Hauff and Georgios Gousios. 2015. Matching GitHub developer profiles to job advertisements. In *Proceedings of the 12th Working Conference on Mining Software Repositories* (MSR '15). IEEE Press, Piscataway, NJ, USA, 362-366.

[6] Dave Binkley, Marcia Davis, Dawn Lawrie, Jonathan I. Maletic, Christopher Morrell, and Bonita Sharif. 2013. The impact of identifier style on effort and comprehension. *Empirical Softw. Engg.* 18, 2 (April 2013), 219-276. DOI=http://dx.doi.org/10.1007/s10664-012-9201-4

[7] David Carmel, Erel Uziel, Ido Guy, Yosi Mass, and Haggai Roitman. 2012. Folksonomy-Based Term Extraction for Word Cloud Generation. *ACM Trans. Intell. Syst. Technol.* 3, 4, Article 60 (September 2012), 20 pages. DOI=http://dx.doi.org/10.1145/2337542.2337545

[8] Dinh Quyen Nguyen and Dinh Dam Le. 2016. Hyper Word Clouds: A Visualization Technique for Last.fm Data and Relationships Examination. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication* (IMCOM '16). ACM, New York, NY, USA, , Article 66 , 7 pages. DOI: http://dx.doi.org/10.1145/2857546.2857613

[9] Eric Alexander and Michael Gleicher. 2016. Assessing Topic Representations for Gist-Forming. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (AVI '16), Paolo Buono, Rosa Lanzilotti, and Maristella Matera (Eds.). ACM, New York, NY, USA, 100-107. DOI: http://dx.doi.org/10.1145/2909132.2909252

[10] Felice Salviulo and Giuseppe Scanniello. 2014. Dealing with identifiers and comments in source code comprehension and maintenance: results from an ethnographically-informed study with students and professionals. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (EASE '14). ACM, New York, NY, USA, , Article 48 , 10 pages. DOI=http://dx.doi.org/10.1145/2601248.2601251

[11] Florian Heimerl, Steffen Lohmann, Simon Lange, and Thomas Ertl. 2014. Word Cloud Explorer: Text Analytics Based on Word Clouds. In *Proceedings of the 2014 47th Hawaii International Conference on System Sciences* (HICSS '14). IEEE Computer Society, Washington, DC, USA, 1833-1842. DOI: http://dx.doi.org/10.1109/HICSS.2014.231

[12] Franz Wanner, Wolfgang Jentner, Tobias Schreck, Andreas Stoffel, Lyubka Sharalieva, and Daniel A Keim. 2015. Integrated visual analysis of patterns in time series and text data – Workflow and application to financial data analysis. Information Visualization, 15, 1, 75–90.

[13] George Fairbanks. 2010. Just Enough Software Architecture: A Risk-Driven Approach. Marshall & Brainerd.

[14] Gregory Moro Puppi Wanderley and Emerson Cabrera Paraiso. 2015. Learning folksonomies from task-oriented dialogues. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing* (SAC '15). ACM, New York, NY, USA, 360-367. DOI: http://dx.doi.org/10.1145/2695664.2695716

[15] Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. Name suggestions during feature identification: the variclouds approach. In *Proceedings of the 20th International Systems and Software Product Line Conference* (SPLC '16). ACM, New York, NY, USA, 119-123. DOI: http://dx.doi.org/10.1145/2934466.2934480

[16] Jared Lorince, Sam Zorowitz, Jaimie Murdock, and Peter M. Todd. 2014. "Supertagger" behavior in building folksonomies. In *Proceedings of the 2014 ACM conference on Web science* (WebSci '14). ACM, New York, NY, USA, 129-138. DOI: http://dx.doi.org/10.1145/2615569.2615686

[17] Ji Wang, Jian Zhao, Sheng Guo, Chris North, and Naren Ramakrishnan. 2014. ReCloud: semantics-based word cloud visualization of user reviews. In *Proceedings of Graphics Interface 2014* (GI '14). Canadian

Information Processing Society, Toronto, Ont., Canada, Canada, 151-158.

[18] Latifa Guerrouj, Massimiliano Penta, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2014. An experimental investigation on the effects of context on source code identifiers splitting and expansion. Empirical Softw. Engg. 19, 6 (December 2014), 1706-1753. DOI=10.1007/s10664-013-9260-1 http://dx.doi.org/10.1007/s10664-013-9260-1

[19] Latifa Guerrouj. 2013. Normalizing source code vocabulary to support program comprehension and software quality. In Proceedings of the 2013 International Conference on Software Engineering (ICSE '13). IEEE Press, Piscataway, NJ, USA, 1385-1388.

[20] Luc Fabresse, Noury Bouraqadi, Christophe Dony, and Marianne Huchard. 2012. A language to bridge the gap between component-based design and implementation. *Comput. Lang. Syst. Struct.* 38, 1 (April 2012), 29-43. DOI=http://dx.doi.org/10.1016/j.cl.2011.10.003

[21] Manamu Sano, Eunjong Choi, Norihiro Yoshida, Yuki Yamanaka, and Katsuro Inoue. 2014. Supporting clone analysis with tag cloud visualization. In *Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices* (InnoSWDev 2014). ACM, New York, NY, USA, 94-99. DOI=http://dx.doi.org/10.1145/2666581.2666586

[22] Mengdie Hu, Krist Wongsuphasawat, and John Stasko. 2017. Visualizing Social Media Content with SentenTree. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (January 2017), 621-630. DOI: https://doi.org/10.1109/TVCG.2016.2598590

[23] Michael Burch, Steffen Lohmann, Daniel Pompe, and Daniel Weiskopf. 2013. Prefix Tag Clouds. In *Proceedings of the 2013 17th International Conference on Information Visualisation* (IV '13). IEEE Computer Society, Washington, DC, USA, 45-50. DOI=http://dx.doi.org/10.1109/IV.2013.5

[24] Michael Burch, Steffen Lohmann, Fabian Beck,Nils Rodriguez, Lorenzo Di Silvestro, and Daniel Weiskopf. "RadCloud: Visualizing Multiple Texts with Merged Word Clouds". 2014. In *Proceedings of the 18th International Conference on Information Visualisation*. Paris, 108-113.

[25] Michael Curtotti, Eric McCreath, and Srinivas Sridharan. 2013. Software tools for the visualization of definition networks in legal contracts. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law* (ICAIL '13). ACM, New York, NY, USA, 192-196. DOI=http://dx.doi.org/10.1145/2514601.2514625

[26] Miltiadis Allamanis and Charles Sutton. 2014. Mining idioms from source code. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (FSE 2014). ACM, New York, NY, USA, 472-483. DOI=10.1145/2635868.2635901 http://doi.acm.org/10.1145/2635868.2635901

[27] Miltiadis Allamanis, Earl T. Barr, Christian Bird, and Charles Sutton. 2015. Suggesting accurate method and class names. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015). ACM, New York, NY, USA, 38-49. DOI=10.1145/2786805.2786849 http://doi.acm.org/10.1145/2786805.2786849

[28] Miltiadis Allamanis, Earl T. Barr, Christian Bird, and Charles Sutton. 2014. Learning natural coding conventions. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (FSE 2014). ACM, New York, NY, USA, 281-293. DOI=10.1145/2635868.2635883 http://doi.acm.org/10.1145/2635868.2635883

[29] Ming-Te Chi, Shih-Syun Lin, Shiang-Yi Chen, Chao-Hung Lin, and Tong-Yee Lee. 2015. Morphable Word Clouds for Time-Varying Text Data Visualization. *IEEE Transactions on Visualization and Computer Graphics* 21, 12 (December 2015), 1415-1426. DOI=http://dx.doi.org/10.1109/TVCG.2015.2440241

[30] Muhammad Ali Babar, Alan W. Brown, and Ivan Mistrik. 2013. *Agile Software Architecture: Aligning Agile Processes and Software Architectures* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[31] Otavio Augusto Lazzarini Lemos, Adriano Carvalho de Paula, Gustavo Konishi, Joel Ossher, Sushil Bajracharya, and Cristina Lopes. 2013. Using Thesaurus-Based Tag Clouds to Improve Test-Driven Code Search. In *Proceedings of the 2013 VII Brazilian Symposium on Software Components, Architectures and Reuse* (SBCARS '13). IEEE Computer Society, Washington, DC, USA, 99-108. DOI=http://dx.doi.org/10.1109/SBCARS.2013.21

[32] Paul Mendoza del Carpio. 2016. Identification of architectural technical debt: An analysis based on naming patterns. In *Proceedings of the 2016 8th Euro American Conference on Telematics and Information Systems (EATIS)*. IEEE Computer Society, Washington, DC, USA, 10-. DOI: http://dx.doi.org/10.1109/EATIS.2016.7520104

[33] Paul Mendoza del Carpio. 2017. A Technique Based on Naming Patterns for Finding Candidates to Components from Source Code. IEEE América Latina, 15, 3 (March 2017).

[34] Paulo Pagliosa, Rafael Messias Martins, Douglas Cedrim, Afonso Paiva, Rosane Minghim, and Luis Gustavo Nonato. 2013. MIST: Multiscale Information and Summaries of Texts. In *Proceedings of the 2013 XXVI Conference on Graphics, Patterns and Images* (SIBGRAPI '13). IEEE Computer Society, Washington, DC, USA, 91-98. DOI=http://dx.doi.org/10.1109/SIBGRAPI.2013.22

[35] Quim Castellà and Charles Sutton. 2014. Word storms: multiples of word clouds for visual comparison of documents. In *Proceedings of the 23rd international conference on World wide web* (WWW '14). ACM, New York, NY, USA, 665-676. DOI: http://dx.doi.org/10.1145/2566486.2567977

[36] Rianne Kaptein. 2012. Effective focused retrieval by exploiting query context and document structure. *SIGIR Forum* 45, 2 (January 2012), 108-108. DOI=http://dx.doi.org/10.1145/2093346.2093366

[37] Rosa Tsegaye Aga and Christian Wartena. 2015. Constructing concept clouds from company websites. In *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business* (i-KNOW '15). ACM, New York, NY, USA, , Article 38 , 4 pages. DOI=http://dx.doi.org/10.1145/2809563.2809615

[38] Surafel Lemma Abebe, Sonia Haiduc, Andrian Marcus, Paolo Tonella, and Giuliano Antoniol. 2009. Analyzing the Evolution of the Source Code Vocabulary. In *Proceedings of the 2009 European Conference on Software Maintenance and Reengineering* (CSMR '09). IEEE Computer Society, Washington, DC, USA, 189-198. DOI=http://dx.doi.org/10.1109/CSMR.2009.61

[39] Stefania Leone, Matthias Geel, and Moira C. Norrie. 2011. The use of tag clouds to support the discovery and inspection of information services. In *Proceedings of the 5th International Workshop on Web APIs and Service Mashups* (Mashups '11), Agnes Koschmider, Erik Wilde, and Christian Zirpins (Eds.). ACM, New York, NY, USA, , Article 10 , 6 pages. DOI=http://dx.doi.org/10.1145/2076006.2076018

[40] Sushil Bajracharya, Joel Ossher, and Cristina Lopes. 2010. Searching API usage examples in code repositories with sourcerer API search. In *Proceedings of 2010 ICSE Workshop on Search-driven Development: Users, Infrastructure, Tools and Evaluation* (SUITE '10). ACM, New York, NY, USA, 5-8. DOI=http://dx.doi.org/10.1145/1809175.1809177

[41] Walid Maalej, Rebecca Tiarks, Tobias Roehm, and Rainer Koschke. 2014. On the Comprehension of Program Comprehension. *ACM Trans. Softw. Eng. Methodol.* 23, 4, Article 31 (September 2014), 37 pages. DOI=http://dx.doi.org/10.1145/2622669

[42] Xiaotong Liu, Han-Wei Shen, and Yifan Hu. 2015. Supporting multifaceted viewing of word clouds with focus + context display. Information Visualization, 14, 2, 168–180.

[43] Yue Yu, Huaimin Wang, Gang Yin, and Tao Wang. 2016. Reviewer recommendation for pull-requests in GitHub. *Inf. Softw. Technol.* 74, C (June 2016), 204-218. DOI=http://dx.doi.org/10.1016/j.infsof.2016.01.004