Visualization of Knowledge Distribution across Development Teams using 2.5D Semantic Software Maps

Daniel Atzberger, Tim Cech, Adrian Jobst,
Willy Scheibel [a], Daniel Limberger [b], Matthias Trapp [c], and Jürgen Döllner

Hasso Plattner Institute, Digital Engineering Faculty, University of Potsdam, Germany
{daniel.atzberger, tim.cech, willy.scheibel, daniel.limberger, juergen.doellner}@hpi.uni-potsdam.de
{adrian.jobst}@student.hpi.uni-potsdam.de

Abstract: In order to detect software risks at an early stage, various software visualization techniques have been developed for monitoring the structure, behaviour, or the underlying development process of software. One of greatest risks for any IT organization consists in an inappropriate distribution of knowledge among its developers, as a projects' success mainly depends on assigning tasks to developers with the required skills and expertise. In this work, we address this problem by proposing a novel Visual Analytics framework for mining and visualizing the expertise of developers based on their source code activities. Under the assumption that a developer's knowledge about code is represented directly through comments and the choice of identifier names, we generate a 2D layout using Latent Dirichlet Allocation together with Multidimensional Scaling on the commit history, thus displaying the semantic relatedness between developers. In order to capture a developer's expertise in a concept, we utilize Labeled LDA trained on a corpus of Open Source projects. By mapping aspects related to skills onto the visual variables of 3D glyphs, we generate a 2.5D Visualization, we call *KnowhowMap*. We exemplify this approach with an interactive prototype that enables users to analyze the distribution of skills and expertise in an explorative way.

1 INTRODUCTION

Visual Analytics for Software Data   The complexity of modern software development projects requires careful coordination of all development activities, which are usually managed using various interactive tools with visualization capabilities. Therefore during the entire development process, data from heterogeneous categories is generated and stored in respective software repositories. For example, changes in the source code made by the developers are represented in a Version Control System (VCS), the progress of development tasks is tracked within an issue tracking system, defects are reported, assigned and managed in a bug tracking system, and discussions on selected topics are conducted in chat forums by the developers. These rich sources of data motivate the use of machine learning techniques when mining software repositories (Bird et al., 2015).

[a] https://orcid.org/0000-0002-7885-9857
[b] https://orcid.org/0000-0002-9111-4809
[c] https://orcid.org/0000-0003-3861-5759

Complementary to the analysis of software data using machine learning techniques, Software Visualization techniques can be applied to communicate various aspects of the structure or behaviour of a software system using graphical representations. Often, the analysis and visualization go hand in hand, therefore being an example for the broader field of Visual Analytics (Keim et al., 2008).

Examples for Software Visualizations supporting program comprehension tasks or risk monitoring include treemaps based on the folder structure of source code projects, where additional quantitative data about a software's quality or complexity is mapped to visual variables (Bohnet and Döllner, 2011; Limberger et al., 2019; Scheibel et al., 2020), circular bundle views for displaying dependencies between source code files (Trümper et al., 2012), trace visualization techniques (Trümper et al., 2010) or software landscapes displaying the semantic structure that is captured from investigating the use of natural language in comments and identifier names using topic models (Kuhn et al., 2008; Atzberger et al., 2021a; Atzberger et al., 2021b).

Figure 1: *KnowhowMap* for the *Bitcoin Core* project(github.com/bitcoin/bitcoin) based on 2000 commits. Each developer is displayed as a pawn figure, whose height displays its expertise level in the concept *Cryptocurrency*. The pointers display latent source code topics.

Problem Statement According to Sommerville (Sommerville, 2016) "The people working in a software organization are its greatest assets", therefore it is of fundamental importance for any IT organization to keep track of the distribution of knowledge across its developer staff. The existing approaches of analyzing knowledge of developers all focus on specific aspects and do thus not allow a user, e.g., a project manager, to investigate the distribution of knowledge in an explorative way. An interactive visualization technique should address the following aspects:

1. **Similarity between developers**, given as a numerical value between 0 and 1, which is the basis for all questions on the distribution of skills.

2. **Visualization of categorical attributes**, as properties, such as the location or the team membership are of discrete nature.

3. **Visualization of numerical attributes**, as the level of skills in a concept or technology can be measured on a numerical scale.

Approach and Contributions In this work, we propose a concept for mining and visualizing the distribution of knowledge across software organizations on the basis of their activities stored in the VCS. By applying Latent Dirichlet Allocation (LDA), a probabilistic topic model, on the commit histories of developers, we can compare developers on a semantic level. A subsequent application of dimension reduction techniques generates a 2D layout for presenting the "similarity" between the developers. Using a novel concept-location approach based on Labeled LDA (LLDA), we assign an expert level to each developer with respect to general concepts, e.g., *Machine Learning* or *Blockchain*. The so-generated data is then mapped onto the visual variables of 3D glyphs, that leads a visualization, we call *KnowhowMap*. An example for a *KnowhowMap* for the *Bitcoin* project is shown in Fig. 1. To summarize, we make the following contributions

1. We present an approach for mining the expertise of developers in general concepts using LLDA.

2. We present a 2.5D visualization technique, whose layout reflects the semantic relatedness between developers and whose glyphs can be used to display various data related to developer data.

The remainder of this work is structured as follows: Sec. 2 describes preliminaries on data sets and concept. Sec. 3 reviews related work with respect to mining developer expertise and visualizing the semantic structure of source code. Our approach for applying LDA and its variant LLDA for mining expertise is described in Sec. 4. Sec. 5 presents our visualization approach. Sec. 6 concludes this paper and outlines directions for future work.

2 BACKGROUND

Application Context   During the entire software development process, a project managers task is to locate expertise in the set of all developers involved in a project in order to assign tasks to the person with the respective skills. Furthermore, it is of major interest for any IT organization to monitor the distribution of its staff as a whole, as an imbalanced concentration of knowledge can lead to significant risks (Cosentino et al., 2015). In large, often globally distributed, teams it is practically impossible to manually keep track of every developers' skill set. But even in small development teams, which are often characterized by an agile and thus dynamic team composition, this is a non-trivial endeavor.

Data Set & Data Acquisition   We focus our analysis on the source code activities of developers, as a large part of a developer's knowledge about code is encoded in it in the form of comments or the choice of identifier names. Thus, mining the commit history of a developer, stored in the VCS, should enable us to draw conclusions about his field of expertise.

Fig. 2 shows the size of the vocabulary for a set of projects from *GitHub* that are associated with the same general concept. The graphs show that only a few representative projects are required until the size of vocabulary stagnates. This shows that projects, with the same underlying concept, share a relatively small vocabulary. Therefore, the similarity between two developers that are active in the same field, is probably captured in their vocabularies.
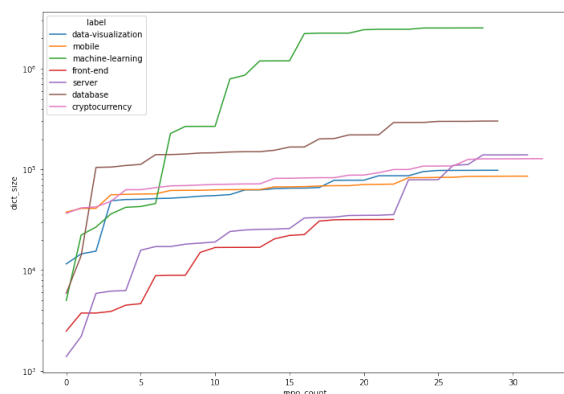

Figure 2: Size of the vocabulary for number of *GitHub* projects that are tagged with the same concept. Each curve flattens after a few projects are added to the set of projects, e.g., for the concept "machine learning" 16 projects are required, and for the concept "database" only 2 projects are required.

Use Cases & Requirements   Our interactive visualization technique addresses two main use cases.

UC-1: The proposed visualization should be able to support a project managers daily work in locating experts for general concepts among the developers. It should facilitate the detection of developers that are active in a general concept, and it should also be able to relate developers to each other, i.e., if a developer leaves a project.

UC-2: On a broader perspective our visualization should be able to monitor the distribution of knowledge across an entire IT organization in order to detect risks at an early stage.

3 RELATED WORK

Our work mainly extends ideas presented by Linstead et al. , who were the first to mine developer expertise using probabilistic topic models (Linstead et al., 2007; Linstead et al., 2009). In particular, a developer's expertise in topics was determined using the Author-Topic Model (ATM), a variant of LDA (Rosen-Zvi et al., 2004), on source code files. In combination with Multidimensional Scaling (MDS), the similarity between the developers can be visualized as a two-dimensional scatter plot.

Saxena and Padanekar proposed an algorithm for mining and visualizing a developer's expertise in abstracts concepts (Saxena and Pedanekar, 2017). By annotating a developers import statements with tags from Q&A-forums, e.g., *Stack Overflow*, each developer was assigned a level-of-expertise in the concept. Based on the underlying hierarchy of concepts, each developer was visualized using a treemap.

Greene and Fisher developed *CVExplorer*, an approach for mining and visualizing technical skills for *GitHub* users (Greene and Fischer, 2016). Their approach extracts the commit messages, the changed files, and the Readme files for each repository a user has contributed to. By comparing the commit messages and the Readme files with a white list of skills, and extracting the programming language from the modified files, each developer is assigned a skill level. The skills for each developer are displayed as an interactive tag cloud.

Kourtzanidis et al. trained the Microsoft Language Understanding Intelligent Service (LUIS) to identify expertise in two front-end frameworks and three .Net technologies (Kourtzanidis et al., 2020). The authors developed a tool, named *RepoSkillMiner*, which visually communicates a developer's profile using basic two-dimensional charts.

Other approaches for mining aspects of developer expertise include familiarity with source code (Fritz et al., 2014), libraries (Teyton et al., 2013), or abstract concepts (Teyton et al., 2014).

Skupin was the first to apply dimension reduction techniques for the task of displaying semantic relatedness between natural language documents (Skupin, 2004). Applying Self-Organizing Maps on the Bag-of-Words (BOW) of abstracts from publications results in a two-dimensional layout that was enriched using cartographic methods.

Kuhn et al. presented a similar method for the software visualization domain, to capture the "semantic" similarity between source code files, i.e., files that implement a common semantic should be placed near each other (Kuhn et al., 2008; Kuhn et al., 2010). Their layout is determined using Latent Semantic Indexing (LSI), a non-probabilistic topic model (Deerwester et al., 1990), together with MDS on the natural language used in the source code files.

Atzberger et al. presented *Software Forest*, a 2.5D visualization, whose layout originates from LDA and MDS applied on the comments and identifier names found in source code files (Atzberger et al., 2021a). By placing 3D glyphs, e.g., trees, on the plane numerical attributes associated to a source code file can be displayed. Their layout approach was later applied to the 3D case, leading to a visualization of development activities in source code using a Galaxy metaphor (Atzberger et al., 2021b).

4 MINING DEVELOPER EXPERTISE FROM SOURCE CODE ACTIVITIES

As discussed in Sec. 2, the vocabulary used in the commits is a strong indicator for a developer's fields of expertise. This motivates the use of techniques from the Natural Language Processing (NLP) domain for mining developer expertise. One of the most widely used class of algorithms for mining software repositories are topic models (Chen et al., 2016). In this section, we detail our approach on how we apply LDA, a probabilistic topic model introduced by Blei et al. (Blei et al., 2003), and its variant LLDA, proposed by Ramage et al. (Ramage et al., 2009), for modelling a developer's expertise based on his source code activities.

Data Preprocessing For each developer, we extract his commit history from the VCS and store the entire set of added and deleted files in a single document.

Before applying the respective topic model, the vocabulary is required to be preprocessed to remove words that carry no meaning and to decrease the size of vocabulary. We adopt the preprocessing steps proposed by Atzberger et al. (Atzberger et al., 2021a),i.e., (1) we remove non-text symbols, e.g., commas or semicolons, (2) we split words at delimiters and according to the Camel Case convention, (3) we remove stop words of the English language and keywords of the programming language, and (4) we lemmatize the vocabulary.

After the preprocessing is performed, we store each document in the BOW format, thus neglecting the order of the single words and just storing their frequency. From now on, when we refer to the commit history of developers, we always mean the preprocessed BOWs.

Latent Dirichlet Allocation for Mining Developer Similarity Given a set of documents $\mathcal{C} = \{d_1, \ldots, d_m\}$, the so-called *corpus*, the goal of a probabilistic topic model is to extract latent topics $\varphi_1, \ldots, \varphi_K$ in the corpus by examining patterns of co-occurring words. The number of topics K is a hyperparameter of the model and needs to be set by the user initially. The extracted topics are given as distributions over the vocabulary $\mathcal{V}$, which contains the terms from the corpus $\mathcal{C}$. In most cases, the underlying "concept" of a topic can be derived from its most probable words (Markovtsev and Kant, 2017). Moreover, topic models learn representations $\theta_1, \ldots, \theta_m$ of the documents as distributions over the topics. The distributions $\theta_1, \ldots, \theta_m$ are therefore descriptions of the semantic structure of the document.

The core assumption of LDA is that the corpus underlies a generative process, which is given by

1. For each document d in the corpus $\mathcal{C}$ choose a distribution over topics $\theta \sim \text{Dirichlet}(\alpha)$

2. For each word w in d

   (a) Choose a topic $z \sim \text{Multinomial}(\theta)$
   (b) Choose the word w according to the probability $p(w|z, \beta)$

By associating each developer with its respective commit history, the application of LDA extracts the hidden concepts, as well as describes each developer as a distribution over topics, i.e., each developer is described as a high-dimensional vector of dimension K.

Locating Concept Expertise using Labeled LDA Direct application of LDA does not provide labels for the extracted topics. Especially when applied on source code files from a single project, the interpretation of topics requires an in-depth knowledge about a

Table 1: Most relevant terms ($\lambda = 0.6$) for six exemplary concepts.

Machine Learning	Mobile	Cryptocurrency	Database	Server	Data Visualization
th	android	fe	err	span	chart
tensor	view	order	db	request	prop
self	name	symbol	table	server	series
cuda	com	crypto	test	test	axis
input	get	binance	key	header	pixi
model	param	price	name	http	gl
license	conv	trade	value	rct	datum
output	activity	wallet	opt	body	react
layer	fpga	exchange	sql	response	point
size	wishlist	block	error	message	style

project (Linstead et al., 2009). Pure LDA is therefore not applicable for locating expertise among developers in general concepts, e.g., *Machine Learning*, *Mobile*, *Front-End*, or *Cryptocurrency*.

LLDA, a variant of LDA proposed by Ramage et al. can be used to extract labeled topics (Ramage et al., 2009). Assuming that each document $d_1, \ldots, d_m$ in the training corpus is provided with tags as meta information, each topic is associated with exactly one tag. Vice versa a document can only have a non-zero value in a topic, when it is provided with the corresponding tag. The idea behind our approach is to generate a corpus of labeled software projects, that serve as representatives for the general concepts. *GitHub Topics*[1] provide a tagging system for projects hosted on *GitHub*. Using the *GitHub API* it is possible to select all projects that are tagged as examples for a general concept. To extract the "real" software projects, we neglect all projects that are additionally equipped with tags from a white list that contains words like, *awesome-list*, *tutorial*, or *interview*. To ensure, that the project is actively maintained, we further filter all projects with less than 20 open or closed issues. The remaining projects are sorted according their *stars*[2] in decreasing order. For example the first four elements of the list of projects for the concept *machine-learning*, are *tensorflow/tensorflow*, *keras-team/keras*, *pytorch/pytorch*, and *scikit-learn/scikit-learn*. To reduce the training time, we reduce the size of the corpus utilizing the second observation discussed in Sec. 2, i.e., the size of the vocabulary for a concept flattens after a few projects. We stop adding documents to a concept, when the vocabulary increases less than 1 % within the last five projects.

After selecting the projects for our training corpus,

we join each source code file of a project to a single document, undertake the preprocessing steps described in Sec. 4 and store them as a BOW. The application of LLDA then leads to a description of concepts of interest as distributions over the vocabulary. However, not always the most probable words are the best indicators for a concept, e.g., words that would occur in all projects very frequently would be assigned a high probability in all topics. In order to increase the interpretability of each topic, we compute the relevance values for each term w in each topic k according to the formula

$$r(w, k | \lambda) = \lambda \cdot \log(\varphi_{w,k}) + (1 - \lambda) \log\left(\frac{\varphi_{w,k}}{p_w}\right), \quad (1)$$

where $\varphi_{w,k}$ means the probability of the word w in topic k and p_w is the marginal probability of the word w in the corpus (Sievert and Shirley, 2014). Tab. 1 shows the most relevant terms, according to a balancing factor of $\lambda = 0.6$ for the concepts *Machine Learning*, *Mobile*, *Cryptocurrency*, *Database*, and *Server*, and *Data Visualization*. The frequency of the 50 most relevant terms for each concept are counted in the commit history of each developer. Based on the frequency, we assign a discrete skill level between one and five to each developer, according to his 20 %-quantile, i.e., a developer who belongs to the 20 % of developers who most often use the relevant words is associated with skill level five.

5 VISUALIZATION APPROACH

Our visualization comprise two main parts: (1) the computation of the layout, displaying semantic relatedness between developers, and (2) the mapping of data to the geometric representations for each developer. The resulting visualization can be categorized as

---

[1] https://github.blog/2017-01-31-introducing-topics/
[2] https://docs.github.com/en/get-started/exploring-projects-on-github/saving-repositories-with-stars
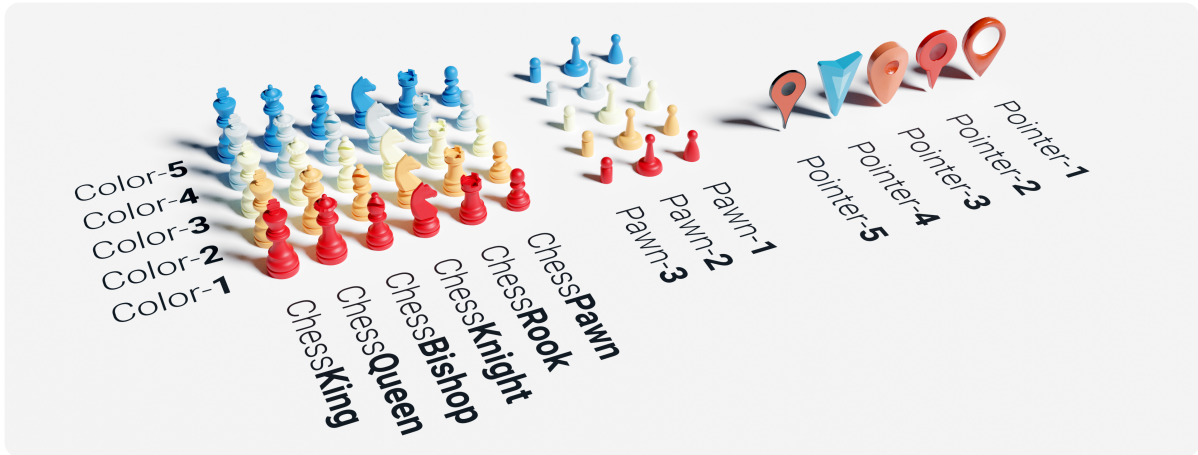
Figure 3: Exemplary atlas of 3D glyphs for representing developers and topics. This atlas uses a diverging 5-color scale for the chess figures and the boardgame pawns. The pointers can be used as landmarks, e.g., for pinpointing the locations of topics.

$\mathcal{A}^3 \oplus \mathcal{R}^2$, i.e., three-dimensional graphical primitives on a two-dimensional reference space (Dübel et al., 2014). By adding interaction techniques, e.g., Zoom, Rotation, and Picking, the user is enabled to explore the map as part of the knowledge mining process.

Layout Technique The objective of our layout computation, is to map the similarity of two developers by Euclidean distance on a two-dimensional reference plane. Although the developers can be compared with each other on the basis of their topic distributions using the Jensen-Shannon distance, the assumption is implicitly made here that all topics are orthogonal to each other, even though topics are themselves distributions over the vocabulary. The layout approach by Atzberger et al. circumvents this problem. In detail, given the latent topics $\varphi_1, \ldots, \varphi_K$ from the corpus of commit histories, their dissimilarities can be measured using the Jensen-Shannon divergence and stored in a square matrix Λ. MDS applied on the dissimilarity-matrix Λ results in K points $\bar{\varphi}_1, \ldots, \bar{\varphi}_K \in \mathbb{R}^2$, such that the Euclidean distance between two points $\bar{\varphi}_i$ and $\bar{\varphi}_j$ represents the entry Λ_{ij} in the dissimilarity matrix (Cox and Cox, 2008). The position for a developer, described as distribution $\theta = \left(\theta^{(1)}, \ldots, \theta^{(K)}\right)$ over the topics, is aggregated as a convex linear combination

$$\bar{\theta} = \sum_{i=1}^{K} \theta^{(i)} \bar{\varphi}_i. \qquad (2)$$

Visual Mapping Our map-related representation shows two types of objects, topics and developers. We decided to use modelled 3D glyphs from *Sketch-Fab*[3] for their representation, as shown in Fig. 3. Our

---
[3]https://sketchfab.com/

system allows the user to manually choose a visual mapping supporting his task.

Topics are presented as pointers, reminding on geographical landmarks. Their purpose is to locate experts in abstracts topics hidden in the source code, support navigating through the set of developers and help creating a mental map. The only visual variable is given by the size and might be accessed for mapping the weight of the topic for the corpus.

Developers are depicted either using pawns or Chess figures. Using a diverging color scheme of five colors, qualitative data can be mapped onto the visual attribute color (Ware, 2019). Furthermore, the type of figure, e.g., *Chess:King* or *Chess:Knight*, can also be used for displaying qualitative information about a developer, e.g., his rank within the software developing. Another visual attribute is the height of each individual figure, which can be used for quantitative data attributes. When we visualize the skill level of a developer, we always make use of the height of each individual glyph.
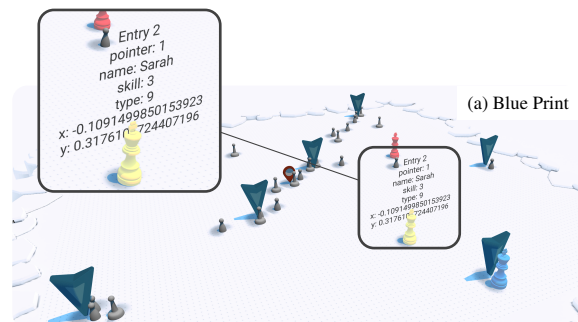

Figure 4: *KnowhowMap* showing a tooltip for a developer to display its position and other attributes.

Implementation Aspects Our data processing pipeline is written in the programming language *Python*. We use the lemmatizer provided by the library *Spacy*[4]. We further use the *nltk* package[5] for accessing the stopwords of the English language. For training an LDA model, we use *Gensim*[6]. The training corpus, consisting of *GitHub* projects, required for training an LLDA model is created using the *GitHub V3 Rest API*. We use the LLDA implementation provided by the Python library *Tomotopy*[7], which implements the algorithm according to the original paper of Ramage et al. (Ramage et al., 2009).

With respect to the visualization prototype, the glyphs are 3D models obtained from *SketchFab*, that are further processed using *Blender*. Each model is associated with a description file (in JSON-format), that specifies the attributes of each object. Our rendering component is based on a *WebGL*-based point-plotter written in *TypeScript* (Wagner et al., 2020) that is able to process large data sets. Further, the prototype supports various interaction techniques, e.g., Zoom, Rotation, and Picking single glyphs. Via hoovering over a glyph, the user can access additional information via a tooltip as shown in Fig. 4. Using the open-source project *webgl-operate*[8], our 2.5D visualization is able to support labeling.

6 CONCLUSIONS

Mining developer expertise is of great interest for industrial software projects, as developers are the most important resource for any IT organization. Existing approaches focus on mining selected aspects of developer expertise, but do not enable the user, e.g., a project manager, to explore the distribution of skills and knowledge in an interactive way.

To address this problem, we propose a Visual Analytics framework for mining and visualizing developer expertise. Each developer is modelled by applying LDA on the commit history of all developers, thus leading to a mathematical formalization. We further train an LLDA model on a dynamically generated corpus of *GitHub* projects to locate expertise in general concepts among developers. The 2D layout of our visualization is derived from LDA and MDS and reflects the semantic similarity between developers and latent topics. We choose 3D glyphs as graphical representation of developers and topics. It is up to the user to choose a mapping between the data related to developer expertise and the visual variables of the glyphs, e.g., color, size and shape.

However, our visualization allows different mappings and it is unclear which mapping is best-suited for each individual task. There, the most important question, that needs to be addressed in future work, is the study of the effectiveness of our approach. We plan to conduct a user study, by interviewing project managers of software projects from industry. We are optimistic that real-world feedback could help to improve our visualization technique.

ACKNOWLEDGEMENTS

REFERENCES

Atzberger, D., Cech, T., de la Haye, M., Söchting, M., Scheibel, W., Limberger, D., and Döllner, J. (2021a). Software forest: A visualization of semantic similarities in source code using a tree metaphor. In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications – Volume 3 IVAPP*, IVAPP '21, pages 112–122. INSTICC, SciTePress.

Atzberger, D., Scheibel, W., Limberger, D., and Döllner, J. (2021b). Software galaxies: Displaying coding activities using a galaxy metaphor. In *Proceedings of the 14th International Symposium on Visual Information Communication and Interaction*, VINCI '21. ACM.

Bird, C., Menzies, T., and Zimmermann, T. (2015). *The art and science of analyzing software data*. Elsevier.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bohnet, J. and Döllner (2011). Monitoring code quality and development activity by software maps. In *Proc. 2nd Workshop on Managing Technical Debt*, MTD '11, pages 9–16. ACM.

Chen, T.-H., Thomas, S. W., and Hassan, A. E. (2016). A survey on the use of topic models when mining software repositories. *Empirical Software Engineering*, 21(5):1843–1919.

---

[4]https://spacy.io/api/lemmatizer

[5]https://www.nltk.org/

[6]https://radimrehurek.com/gensim/

[7]https://bab2min.github.io/tomotopy/v0.12.2/en/

[8]https://github.com/cginternals/webgl-operate

Cosentino, V., Izquierdo, J. L. C., and Cabot, J. (2015). Assessing the bus factor of git repositories. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 499–503.

Cox, M. A. and Cox, T. F. (2008). Multidimensional scaling. In *Handbook of Data Visualization*, pages 315–347. Springer.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dübel, S., Röhlig, M., Schumann, H., and Trapp, M. (2014). 2d and 3d presentation of spatial data: A systematic review. In *Proc. VIS International Workshop on 3DVis*, 3DVis '14, pages 11–18. IEEE.

Fritz, T., Murphy, G. C., Murphy-Hill, E., Ou, J., and Hill, E. (2014). Degree-of-knowledge: Modeling a developer's knowledge of code. *ACM Trans. Softw. Eng. Methodol.*, 23(2).

Greene, G. J. and Fischer, B. (2016). Cvexplorer: Identifying candidate developers by mining and exploring their open source contributions. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 804—809.

Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G. (2008). Visual analytics: Definition, process, and challenges. In *Information visualization*, pages 154–175. Springer.

Kourtzanidis, S., Chatzigeorgiou, A., and Ampatzoglou, A. (2020). Reposkillminer: Identifying software expertise from github repositories using natural language processing. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1353–1357.

Kuhn, A., Erni, D., Loretan, P., and Nierstrasz, O. (2010). Software cartography: Thematic software visualization with consistent layout. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(3):191–210.

Kuhn, A., Loretan, P., and Nierstrasz, O. (2008). Consistent layout for thematic software maps. In *Proc. 15th Working Conference on Reverse Engineering*, WCRE '08, pages 209–218. IEEE.

Limberger, D., Scheibel, W., Döllner, J., and Trapp, M. (2019). Advanced visual metaphors and techniques for software maps. In *Proc. 12th International Symposium on Visual Information Communication and Interaction*, VINCI '19, pages 11:1–11:8. ACM.

Linstead, E., Bajracharya, S., Ngo, T., Rigor, P., Lopes, C., and Baldi, P. (2009). Sourcerer: mining and searching internet-scale software repositories. *Data Mining and Knowledge Discovery*, 18(2):300–336.

Linstead, E., Rigor, P., Bajracharya, S., Lopes, C., and Baldi, P. (2007). Mining eclipse developer contributions via author-topic models. In *Proc. 4th International Workshop on Mining Software Repositories*, MSR '07, pages 30:1–4.

Markovtsev, V. and Kant, E. (2017). Topic modeling of public repositories at scale using names in source code. *arXiv CoRR cs.PL*.

Ramage, D., Hall, D., Nallapati, R., and Manning, C. D. (2009). Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 248–256.

Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494. AUAI Press.

Saxena, R. and Pedanekar, N. (2017). I know what you coded last summer: Mining candidate expertise from github repositories. In *Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 299–302.

Scheibel, W., Trapp, M., Limberger, D., and Döllner, J. (2020). A taxonomy of treemap visualization techniques. In *Proc. 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications – Volume 3: IVAPP*, IVAPP '20, pages 273–280. INSTICC, SciTePress.

Sievert, C. and Shirley, K. (2014). Ldavis: A method for visualizing and interpreting topics. In *Proc. Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70. ACL.

Skupin, A. (2004). The world of geography: Visualizing a knowledge domain with cartographic means. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5274–5278.

Sommerville, I. (2016). *Software Engineering*, volume 10th Edition. Pearson Education.

Teyton, C., Falleri, J.-R., Morandat, F., and Blanc, X. (2013). Find your library experts. In *2013 20th Working Conference on Reverse Engineering (WCRE)*, pages 202–211.

Teyton, C., Palyart, M., Falleri, J.-R., Morandat, F., and Blanc, X. (2014). Automatic extraction of developer expertise. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 1–10. ACM.

Trümper, J., Bohnet, J., and Döllner, J. (2010). Understanding complex multithreaded software systems by using trace visualization. In *Proceedings of the 5th International Symposium on Software Visualization*, SOFTVIS '10, pages 133—142. Association for Computing Machinery.

Trümper, J., Beck, M., and Döllner, J. (2012). A visual analysis approach to support perfective software maintenance. In *2012 16th International Conference on Information Visualisation*, pages 308–315. IEEE.

Wagner, L., Scheibel, W., Limberger, D., Trapp, M., and Döllner, J. (2020). A framework for interactive exploration of clusters in massive data using 3d scatter plots and webgl. In *Proceedings of the 25th International Conference on 3D Web Technology*, Web3D '20, pages 31:1–2. ACM.

Ware, C. (2019). *Information visualization: perception for design*. Morgan Kaufmann.