A Data-Centric Study of Software Tutorial Design

Deeksha M. Arya*, **Mathieu Nassif***, **Martin P. Robillard**
deeksha.arya@mail.mcgill.ca, mnassif@cs.mcgill.ca, martin@cs.mcgill.ca
*These authors contributed equally

***Abstract*—We investigate three tutorials on Android development to elicit design choices related to their organization and content. We analyze organization styles related to the intended reading sequence, use of code fragments, and reliance on external resources. We survey design variations in the choice of topics covered by tutorials, for example deprecated or convenience APIs, and concerns unrelated to the functional features such as development methodologies and best practices. We provide insights about the impact of different design decisions on readers. For example, a clear reading order provides an easier introduction to the technology for beginners, but focused, independent sections can be more effective for an intermediate-level audience. We describe these decision points and their trade-offs and propose guidelines to assist tutorial authors in making explicit and informed decisions about the design of tutorials.**

■ **Developers routinely refer to** introductory tutorials when learning new APIs [9]. Effective tutorials are thus a benefit to the technology learners. In turn, they benefit organizations that develop and promote the use of the technology by raising community interest and adoption levels [3].

How the design of tutorials can support effective learning, however, remains an open question. A glance at the offerings for popular technologies shows a diversity of styles related to content selection and presentation. To better understand the design dimensions for software tutorials, we conducted a systematic data-centric analysis of three tutorials aimed at beginners to Android programming: The App Basics tutorial from the official Android Developer Documentation (AndroidOfficial), Google's Android Developer Fundamentals tutorial (GoogleCourse), and Vogella's Android Development tutorial (VogellaTraining) (see *Tutorials* side bar).

We selected these tutorials because of their cohesiveness, up-to-dateness, and authoritativeness. We avoided the many tutorials that consist of loosely structured collections of poorly edited blog posts. We used this selection process to elicit a variety of design decisions made by professional writers, rather than to gather a representative sample of tutorials. The systematic approach by which we performed this selection is available in our online appendix at https://zenodo.org/record/4276356.

Despite their similar audience and purpose, the tutorials differ in essential ways. We observed radical variations in their division into sections and their use of code fragments and links to other documentation. The overlap of topic coverage between tutorials was also surprisingly low, hinting at different content selection strategies.

Side Bar: Tutorials
AndroidOfficial is the set of introduction documents under the "App Basics" header from the Android platform [4]. It is the of-

ficial set of short references to get programmers started quickly on Android development. *GoogleCourse* is the "Android Developer Fundamentals" tutorial created by the Google Developers Training Team to accompany an in-person course leading to an entry-level Android certification [6]. *VogellaTraining* is the set of training resources under the "development starter" and "fundamental" sections of the Android Development tutorials of the Vogella training platform [10].

Design Decisions

We compared the *organization* and *content* of the three tutorials, following Van der Meij et al.'s discussion of the evolution of content and presentation of software tutorials between 1980 and 2009 [8]. We observed several impactful differences between tutorials, which we articulate as eleven design decisions along five dimensions. Table 1 summarizes these decisions, with their rationales and trade-offs. Importantly, different parts of the same tutorial can realize different decisions.

For example, a tutorial creator hired to document new features of an API may choose to design the tutorial as a *modular* set of independent sections for an audience already familiar with the technology. They may use *focused code fragments* throughout each section to keep the focus on the new features, and base the content of the tutorial on *external factors* such as popular feature requests to showcase the value of the updated API.

In the remainder of this article, we discuss each dimension in depth, including examples of design variations from the three Android tutorials, and details of the process we followed to analyze the tutorials.

Tutorial Organization

Table 2 reports structural properties of the tutorials under study. Although the three tutorials are not intended to be representative of all tutorials, we compared them to eleven other introductory Android tutorials to assess how they fit among the range of available tutorials (see the online appendix for the details of our sampling strategy). We found that, except for an uncharacteristically high number of hyperlinks, they do not exhibit unusual properties.

Intended Reading Order

All three tutorials are designed to be read in different manners. GoogleCourse's content is organized in a single sequence to read in a prescribed order. This single sequence is easy to follow for beginners who may not know in advance what information is the most relevant.

In contrast, AndroidOfficial does not have a clear reading order. Each document contains only information related to a focused subject, and delegates related information to other documents. Thus, AndroidOfficial consists of a complex network of 56 short documents (with 5.2 sections on average, compared to 21.1 and 33.3 sections for GoogleCourse and VogellaTraining) linked by 529 references to each other. Decoupled documents can improve their reusability in other learning frameworks, according to Boyle [2]. They are also useful to readers with specific information needs: Readers can access the desired information without wasting time on context built in prior documents. Links to other documents provide learning resources for related concepts if necessary.

VogellaTraining, with its ten documents, lies between AndroidOfficial and GoogleCourse with regard to its reading sequence. The largest document (103 sections and 8437 words, more than twice the size of the second largest) covers a list of concepts that beginners should read in sequence, similarly to GoogleCourse. The other nine documents explore in more details different fundamental aspects of Android. These shorter documents share similarities with AndroidOfficial's documents, as they can be read in any order and focus on a specific concept. The combination of both organization styles is a compromise that grants readers the flexibility to explore different aspects of the framework as they please, after having been introduced to fundamental notions relevant to all of these aspects.

Use of Code Fragments

The tutorials exemplify two approaches for presenting code fragments to the reader.

Table 1. Design dimensions, with sample decisions and their impact on the readers.

	Decision	When/Why	Trade-Off
Organization	*Structure of the tutorial components*		
	Sequential sections forming a single narrative	Provides beginners with an explicit entry point and a measurable progression	Requires readers to go through sections that may be less relevant to them
	Modular set of independent, focused sections	Allows more advanced developers to only read sections that address their information needs	Creates more complex dependencies between sections that can be challenging to navigate
	Context included in a code fragment		
	Complete **compilable code** (e.g., entire files)	Encourages readers to clone the examples for a more participatory tutorial	Can distract from or hide the code elements of interest
	Short **focused fragments** (e.g., few statements)	Focuses the discussion on relevant code, e.g., when comparing different approaches	Can become challenging for readers to integrate many focused fragments together
	Links to external resources		
	Integral component of the tutorial	Reduces tutorial creation and maintenance effort, and provides a broad overview of the topic	*All links:* Each additional link can distract the reader, who needs to jump back and forth between the tutorial and external resources
	Optional supplement for specialized topics	Allows readers to further their expertise and interest on a topic	
Content	*Main topic selection strategy*		
	Selection based on **external factors**	Tailors the tutorial content to demonstrated information needs	Reduces the cohesiveness of topics, compared to a baseline reflecting the author's perspective
	Selection based on **interactions** between topics	Describes how topics work together to build more complex applications	Requires a lot of effort: the number of interactions grows exponentially with topics
	Selection of additional topics		
	Implementations of broad core topics	Provides concrete details to understand a core topic in a specific context	*All topics:* Each additional topic lengthens the tutorial, making it more expensive to create and maintain, and more daunting to readers who may want to get coding quickly
	Non-functional topics (e.g., development tools)	Introduces beginners to good development practices early	
	Peripheral topics (e.g., third-party libraries, deprecated APIs)	Addresses varying needs of readers working on specialized applications or legacy systems	

AndroidOfficial and GoogleCourse mostly contain short code fragments that focus on the statements of interest. In contrast, VogellaTraining's code fragments often display an entire file, including more trivial information such as the package and import declarations. Despite these general tendencies, all three tutorials use both short and long code fragments at some point, with the largest code fragment having 2110, 1525, and 2368 characters respectively for AndroidOfficial, GoogleCourse, and VogellaTraining.

Code fragments that focus on a single method do not overwhelm the reader with unnecessary information. They convey a clearer purpose. In contrast, showing the complete content of a file provides context for the relevant code. It thereby allows readers to follow the evolution of code through several manipulations and understand how different concepts interact in a complete application.

Another interesting design decision, although only observed in AndroidOfficial, is to present equivalent code fragments in both languages officially supported for Android development, Java and Kotlin. The tutorial uses a tabbing mechanism to show fragments in the language the reader prefers. This design can increase the audience of a tutorial, but requires the additional cost of creating and

Table 2. Properties of the three tutorials. A document refers to a web page, delimited into sections by a header (HTML h1-h3 tags). We also compare our subjects to eleven other Android tutorials for context.

Property	Android-Official	Google-Course	Vogella-Training	Others (11)		
				Min.	Mean	Max.
Documents	56	33	10	4	38	109
Sections	292	697	333	16	252	712
Words	83 351	103 431	21 890	1559	30 925	104 441
... per document	1488	3134	2189	212	948	2984
... per section	285	148	66	32	135	240
Code fragments	338	430	174	2	222	711
Visible characters in code fragments	67 810	90 107	102 141	172	132 528	454 856
... per code fragment	201	210	587	86	607	905
Hyperlinks (excluding self-referencing)	1447	1800	64	3	237	847
... to other pages of the tutorial	529	2	4	0	42	124
... to advanced tutorial pages	110	0	2	1	92	340
... to API reference documentation	602	1018	0	0	4	29
... to other resources	206	780	58	1	98	606

maintaining pairs of equivalent code fragments.

Links to External Resources

Resources found outside a tutorial can complement the content of tutorials. These external resources can include pages of advanced tutorials from the same website, official API reference documentation, and other resources, for example, blogs and third party libraries.

AndroidOfficial uses external resources to lighten its content, allowing readers to go through each document more quickly. It contains 110 links to advanced tutorial pages hosted on the same website. There are 602 links to API reference documentation in AndroidOfficial, to avoid redundant descriptions of API types. It also contains 206 links to other resources, among which 163 links refer to official documentation hosted on the Android Developer website, such as graphical design guides. This large number of links is representative of the Android documentation website, which can be viewed as a large network of learning resources, of which AndroidOfficial is a subset. These external resources conveniently refer readers to additional concepts, but can break the flow of the tutorial if readers navigate back and forth between the tutorial and external resources.

GoogleCourse also contains many links (1798) to external resources. The 1018 links to API reference documentation are due to mentions of API types being systematically linked to their reference documentation. The remaining 780 links point to a variety of resources, including websites of different technologies (e.g., the Mockito framework), Stack Overflow posts, and coding exercises, often under a “Learn more” header. This use of links contrasts with AndroidOfficial, as external resources are explicitly marked as supplemental material. Hence, in GoogleCourse, the external resources complement, rather than directly support, the content of the tutorial.

VogellaTraining contains far fewer links to external resources, amounting to 60 in total. It contains zero references to the API documentation, and so often repeats text that is already present in the official documentation. Without a method to synchronize updates to the official documentation with the content in the tutorial, VogellaTraining faces the risk of containing inconsistent information and becoming outdated [1]. The hyperlinks that it does contain usually point to the root page of documentation-hosting websites rather than specific documents. Similarly to GoogleCourse, these few links in the main text of the tutorial encourage readers to remain within the tutorial until its completion to limit potential distractions. VogellaTraining even includes programming exercises between the more conceptual sections, limiting the need for readers to refer to an external resource for practice material.

Tutorial Content

The lack of content presentation standard for tutorials makes their investigation a difficult problem [5]. To provide an objective definition of a *tutorial content topic*, we used Stack Overflow tags as proxies for topics.

We retrieved tags that contain the substring "android". After removing irrelevant tags, such as specific Android versions, we obtained 393 topics, related to API types (e.g., android-intent), libraries (e.g., android-glide), and generic concepts (e.g., android-camera).

For each of the $3 \times 393 = 1179$ tutorial–topic pairs, we manually identified whether the tutorial covers the topic, reporting the section where the topic is covered (when applicable) as evidence. We did not discriminate between degrees of coverage, but considered passing mentions as insufficient to cover a topic. All authors independently annotated a distinct set of pairs which included a common subset of 20 tutorial–topic pairs from each tutorial. On this common subset, the annotators achieved a pairwise agreement of 95% to 100% (Cohen's κ between 0.83 and 1.00) for AndroidOfficial and GoogleCourse, and of 80% to 100% (Cohen's κ between 0.49 and 1.00) for VogellaTraining, which demonstrates the reasonably low subjectivity of the task. The resulting mapping between topics and tutorials, with further methodological details, are available as part of our online appendix.

This procedure produced a coverage incidence matrix with three rows (tutorials) and 393 columns (topics), which provides valuable insights into different strategies to select the content of a tutorial. Figure 1 shows this incidence matrix (middle section), in relation with the popularity of each topic (top section).

Core and Additional Topics

We expected tutorials to cover a common core of essential Android topics. However, among the 393 topics, only 36 are covered by all three tutorials. This intersection of topics among tutorials is small in comparison with the 184 topics covered by at least one tutorial (20%). The 148 topics covered by one or two tutorials are not equally distributed: GoogleCourse covers 119 of them, while AndroidOfficial covers only 32 of these additional topics.

The topics common to all three tutorials include the most basic aspects of Android development. For example, it includes android-layout, android-view, android-activity, and android-intent—the four pillars of any Android app. It also includes android-ide, android-gradle-plugin, and android-emulator to teach beginners to implement, build, and run an application. Thus, all three tutorials cover at least the essential topics to develop a basic application.

Topics covered only by AndroidOfficial relate to features of Android devices, even if they may not be used by a majority of beginners, such as android-gps, and android-strictmode. In contrast, the many topics solely covered by GoogleCourse include convenient API classes (e.g., android-pendingintent) and third-party libraries (e.g., pocketsphinx-android), helping beginners explore topics beyond those strictly necessary to start a project.

VogellaTraining covers some deprecated APIs, such as the popular tag android-listfragment. The coverage of deprecated APIs can be helpful for developers joining older projects or if the classes are used in Android projects despite the deprecation notice. A survey conducted by Lethbridge et al. revealed that 81% of the 45 respondents agreed that even though it may not be up to date, software documentation can still be useful [7].

The inclusion in a tutorial of topics beyond the essential core to use the framework constitutes a design trade-off. Comprehensive tutorials that cover a broader range of topics require additional effort from the authors and may discourage readers by their increased length. However, omitted topics can prevent beginners from exploiting useful features.

Popularity of Topics

We use the number of Stack Overflow questions associated with each topic as a measure of how prevalent its related information needs are among Stack Overflow users. The incidence matrix in Figure 1 shows that tuto-

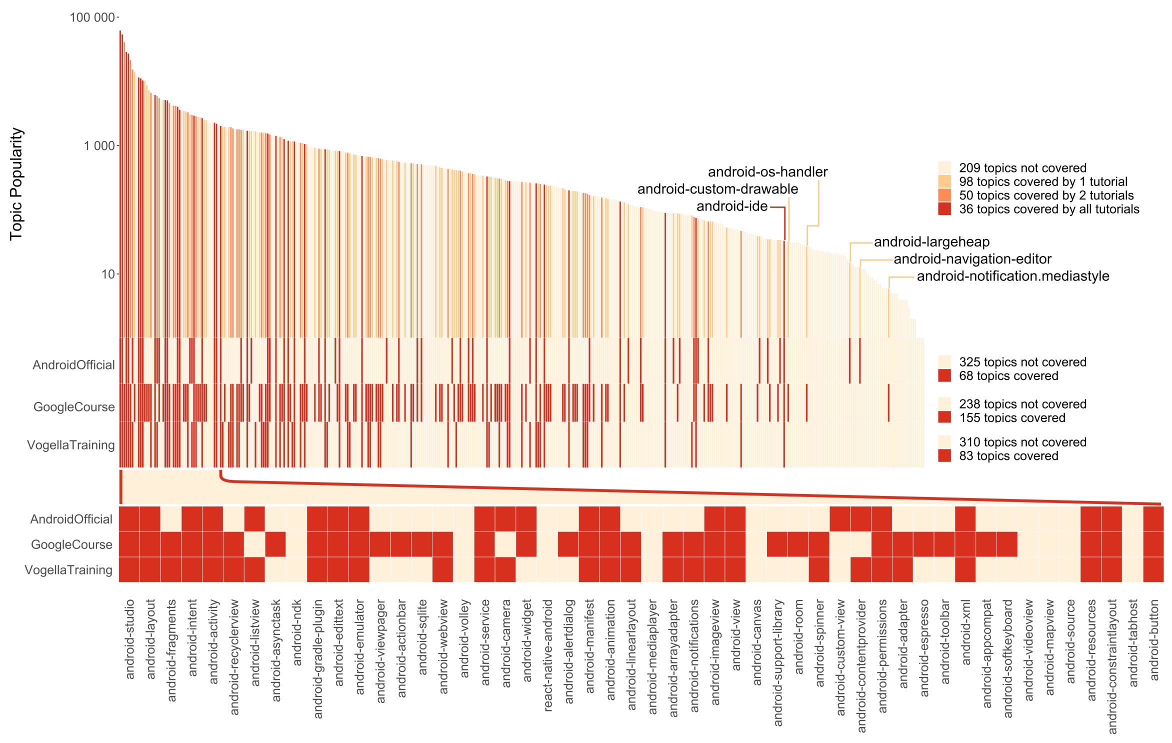


Figure 1. Topics covered by each tutorial. The top section shows the popularity of each topic, measured by the number of Stack Overflow posts tagged by the topic. The middle section indicates, for each 393 topics, which tutorial covers it. The bottom section zooms in on the 50 most popular topics.

rials tend to cover popular topics. AndroidOfficial, GoogleCourse, and VogellaTraining cover respectively 22, 37, and 28 of the 50 most popular topics, but only zero, one, and two of the 50 least popular ones. However, all three tutorials cover topics across the whole range of popularity, and leave out some of the most popular topics that are demonstrably challenging for developers.

AndroidOfficial covers fewer of the popular topics than the other two tutorials. For example, it covers android-view, but not its more popular subclass, android-recyclerview. It excludes other popular topics that are not essential for building applications, such as android-fragments (third most popular). Thus, AndroidOfficial remains strict in its goal to provide the minimal information to build simple applications, as opposed to GoogleCourse and VogellaTraining, which are broader.

Tutorial authors cannot only rely on a topic's popularity to decide whether to cover it, as popularity is not a perfect assessment of relevance. Popularity depends also on the complexity of the topic: prevalent but trivial topics are less likely to generate questions from developers. For example, retrieving metadata about an Android application using the ApplicationInfo class is not very complex, but is an important task for beginners to learn. Introductory tutorials still need to cover these prevalent topics, so a lack of popularity does not indicate that the topic is irrelevant. Some of the least popular topics are covered by all three tutorials, including for example android-applicationinfo, which is among the 150 least popular topics.

Categories of Topics

Many of the covered topics are associated with an API type from the Android Framework (e.g., android-asynctask corresponds to the type android.os.AsyncTask). Although such type-related topics amount to only 34% of all 393 topics, they constitute the majority of topics covered by GoogleCourse and VogellaTraining (50% and 52%, respectively), and

38% of the topics covered by AndroidOfficial. The tutorials also cover many topics related to the architecture and components of Android, e.g., android-styles and android-manifest. Similarly to type-related topics, these topics reveal that GoogleCourse and VogellaTraining largely cover the functionality and behavior of the API, as opposed to other relevant concerns such as third-party libraries and development tools.

Another common kind of covered topics are Android features visible to end-users, such as android-sharing, and android-orientation. For example, AndroidOfficial, GoogleCourse, and VogellaTraining uniquely cover android-keypad, android-launcher, and android-button respectively. We observed that this kind of topic was especially prevalent in topics uniquely covered by AndroidOfficial.

Contrary to AndroidOfficial and VogellaTraining, GoogleCourse covers external libraries such as android-espresso and android-glide. Although it defers to their documentation for a more extensive description, a short introduction increases awareness of the Android development ecosystem.

Finally, all three tutorials cover few topics related to development methods and tools, such as android-lint and android-monkey, ignoring even official development tools such as android-ndk and android-jetpack. This general bias against development and maintenance concerns is surprising, as beginners would benefit from learning good development practices.

Correlated Topics

As software technology is comprised of multiple interacting components, cohesive tutorials cannot cover topics in isolation. For example, Android *activities* and *intents* should be discussed together, as intents are used to switch between activities. To assess the cohesiveness of the tutorial contents, we looked at pairs of topics that frequently co-occur, i.e., are tagged on the same Stack Overflow question. We considered the pairs of topics that co-occur in at least 40 questions, which cumulatively constitute over 75% of all co-occurrences.

Figure 2 shows how many topics of each pair are covered by each tutorial. Both AndroidOfficial and VogellaTraining cover only one topic in almost half of the correlated pairs. Covering the other topic in these pairs would convey a more cohesive and complete perspective of Android programming, but each additional topic can in turn create a new correlated pair with only one covered topic. Tutorial authors must carefully choose when to stop adding related topics to avoid overwhelming a reader. Consistently with the previous observation of a small set of topics covered by all tutorials, only 42 topic pairs (6%) are covered by all three tutorials.

GoogleCourse and VogellaTraining both cover the two topics of *all* but three of the top 20 most frequently co-occurring pairs. In contrast, AndroidOfficial covers only half of the 20 most common pairs. This observation is consistent with the organization of AndroidOfficial as short, decoupled sections. In many cases, when a pair consists of a broad topic and a more specific one, e.g., android-layout with android-linearlayout, AndroidOfficial typically only discusses the broader topic. So AndroidOfficial provides an overall introduction to the framework, but leaves out specific instances of the different concepts.

Towards a Systematic Approach to Tutorial Design

The design of tutorials has evolved over time, for example, by introducing minimalist documentation in response to work on improved usability and readability of shorter, focused texts [8]. This evolution is similar to the evolution of software design, which is supported by conceptual frameworks and tools to systematically assess and document design decisions. However, contrary to software design, no such system exists for tracking tutorial design rationale.

Our investigation of the organization and content of three introductory Android tutorials has revealed many variation points in tutorial design. We find that tutorial creators must be careful in the design decision they make, sometimes unconsciously, as they impact how different audience will receive the tutorial. Even carefully composed tutorials

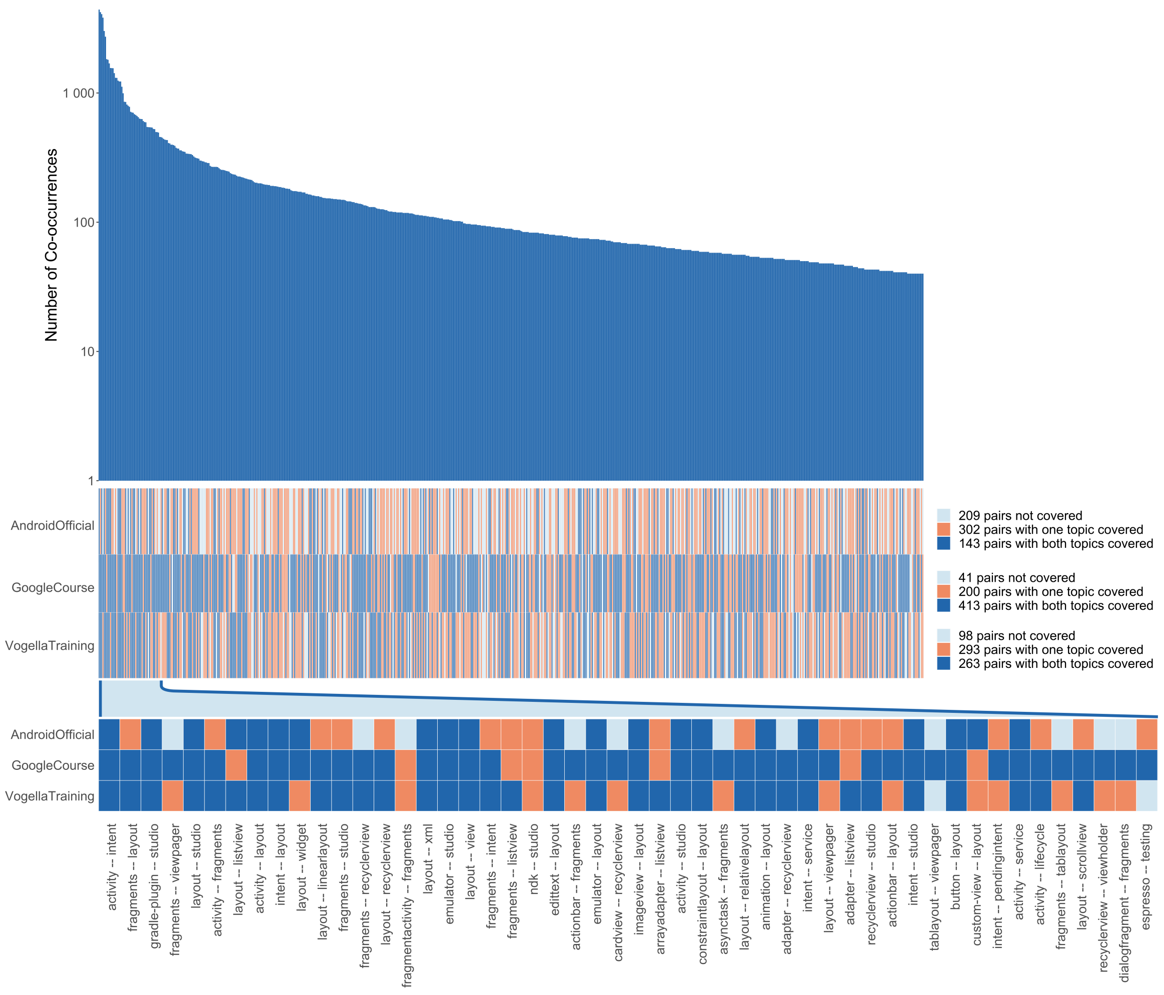


Figure 2. Topics pairs covered by each tutorial, showing only the 654 most common pairs, accounting for over 75% of all occurrences. The top section shows the popularity of each pair, the middle section indicates how many topics of each pair is covered by each tutorial, and the bottom section zooms in on the 50 most frequent pairs (stripping the android- prefix of each topic.)

can be poorly received if its content does not match the information needs of its target audience. Thus, we propose a framework, captured in Table 1, that authors can use to systematically review the design of their tutorials.

To illustrate how these guidelines can be useful, we consider the case of a tutorial creator who is tasked to write a "Getting Started" tutorial to present a novel unit testing library. Their initial draft consists of a series of sections that cover all features of the library, with one feature per section.

Upon reviewing our guidelines, the tutorial creator may realize that their long draft can discourage readers. Thus, they keep only the description of the *core features* in a *sequential* reading order for a quicker introduction. To cater to advanced users, they can refactor sections about *specialized features* into optional *independent* sections, excluded from the main tutorial, and add "Additional Reading" boxes at relevant places in the tutorial to *link* to those sections. They can also decide to cover additional topics based on *external factors*, such as common testing patterns and features of competing libraries. However, to keep the main tutorial more cohesive, they can choose to only place those topics in the optional sections.

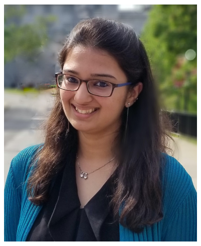
While incorporating our guidelines in Table 1, authors must balance trade-offs based on the context and expectations of each tutorial. In the previous example, the tutorial writer may have instead opted to avoid links to optional sections altogether, to avoid possible distractions for beginners and reduce the tutorial creation effort. Both strategies have their merits. Although our study focused on tutorials targeted at an audience of Android programming beginners, our guidelines are applicable to different technologies and audience expertise levels. The guidelines are thus offered as a tool to stimulate additional reflection and encourage a systematic and informed approach to tutorial design.

Acknowledgements

We are grateful to the participants of the Winter 2020 seminar course on Topics in Mobile Application Development at McGill University for contributing to a pilot study of this project. This work is funded by NSERC.

REFERENCES

1. Deeksha M. Arya, Jin L. C. Guo, and Martin P. Robillard. Information correspondence between types of documentation for APIs. *Empirical Software Engineering*, 25(5):4069–4096, 2020.
2. Tom Boyle. Design principles for authoring dynamic, reusable learning objects. *Australasian Journal of Educational Technology*, 19(1):46–58, 2003.
3. Barthélémy Dagenais and Martin P. Robillard. Creating and evolving developer documentation: Understanding the decisions of open source contributors. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, page 127–136, 2010.
4. Android Developers. Developer guides. https://developer.android.com/guide. Accessed: 2020-06-03.
5. Adam Fourney and Michael Terry. Mining Online Software Tutorials: Challenges and Open Problems. In *Proceedings of Extended Abstracts on Human Factors in Computing Systems*, pages 653–664, 2014.
6. Google. Android developer fundamentals. https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/index.html. Accessed: 2020-06-03.
7. T. C. Lethbridge, J. Singer, and A. Forward. How software engineers use documentation: the state of the practice. *IEEE Software*, 20(6):35–39, 2003.
8. Hans Meij, Joyce Karreman, and Michaël Steehouder. Three decades of research and professional practice on printed software tutorials for novices. *Technical Communication*, 56, 08 2009.
9. Michael Meng, Stephanie Steinhardt, and Andreas Schubert. Application programming interface documentation: What do software developers want? *Journal of Technical Writing and Communication*, 48(3):295–330, 2018.
10. Vogella. Android development tutorials. https://www.vogella.com/tutorials/android.html. Accessed: 2020-06-03.



Deeksha M. Arya is a Ph.D. student in Computer Science at McGill University. She is interested in the software documentation landscape and its role in the education of software technologies for beginners. Deeksha completed a M.Sc. in Computer Science from McGill University in 2019. In her thesis, she systematically analyzed the semantics between corresponding sentences in API reference documentation and tutorials. Her work provides insight on the relatedness of the two documentation types and informs technology to support this correspondence to maintain information consistency. In 2014, Deeksha received a B.E. in Information Science from the M.S. Ramaiah Institute of Technology. Contact her at deeksha.arya@mail.mcgill.ca.

Mathieu Nassif is a Ph.D. student in Computer Science at McGill University. His research focuses on the extraction, representation, and manipulation of knowledge in software systems to optimize the contribution of developers. Mathieu received a M.Sc. in Computer Science from McGill University in 2018. His thesis explored a flexible approach to embed documentation directly in source code to reduce the redundancy of information in software systems while improving documentation quality. Mathieu received a B.Sc. in Mathematics from Université de Montréal in 2016. Contact him at mnassif@cs.mcgill.ca.

Martin P. Robillard is a Professor of Computer Science at McGill University. His research investigate how to facilitate the discovery and acquisition of technical, design, and domain knowledge to support the development of software systems. He served as the Program Co-Chair for the 20th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2012) and the 39th ACM/IEEE International Conference on Software Engineering (ICSE 2017). He received his Ph.D. and M.Sc. in Computer Science from the University of British Columbia and a B.Eng. from École Polytechnique de Montréal. Contact him at martin@cs.mcgill.ca.