A Field Study of Developer Documentation Format

Mathieu Nassif
mnassif@cs.mcgill.ca
McGill University
Montréal, Canada

Martin P. Robillard
martin@cs.mcgill.ca
McGill University
Montréal, Canada

ABSTRACT

Documentation facilitates the transfer of knowledge among programmers and helps them become familiar with new technologies. However, the effectiveness with which a reader can find information in a document depends on its presentation format. In a prior publication, we presented Casdoc, a novel dynamic format for code examples. In this work, we synthesized five documentation presentation guidelines from prior research on programmer information needs and search behaviors. We then used Casdoc as an instrument to evaluate the impact of these guidelines in a field study with 326 students who used 126 documents over several months. Participants overwhelmingly chose to use Casdoc instead of a static baseline format. We observed that interactive documents can contain more information without distracting its readers. We also found some limitations that authors should consider when applying the guidelines, such as the large impact of small differences in visual cues to help readers navigate a document.

CCS CONCEPTS

- Software and its engineering → Documentation; - Social and professional topics → Computer science education; - Human-centered computing → Field studies; - Web-based interaction; - Interactive systems and tools.

KEYWORDS

field study, software documentation, documentation format, dynamic documentation, code examples

ACM Reference Format:

1 INTRODUCTION

Documentation is a major asset to programmers learning a new software system. This importance has been recognized by researchers, who proposed many techniques to generate documents (e.g., [9, 13, 14, 26, 30]). However, high-quality documentation must not only contain sufficient relevant information to address the needs of its consumers: it must also be presented in a way that allows consumers to effectively find that information. As documentation becomes more complex and addresses the needs of a larger audience, its usefulness increasingly relies on an effective presentation format to avoid common issues such as appearing too verbose or technical, or lacking background information [2].

Despite the importance of a document’s format on its quality, the effectiveness of different presentation styles is not well understood. Technical writers use various strategies to present information [4], but without empirical insights to guide their decisions. Thus, they risk wasting efforts authoring documents that are ultimately undermined by an ineffective format.

To improve software documentation practices, we investigated the impact of presentation features on documentation quality. We focused on interactive presentation features to deliver tailored content to a wide audience, as past work has demonstrated the value of interactivity in online documents [35, 37, 48]. Our investigation focuses on the presentation of code examples in tutorial resources. We synthesized five presentation guidelines from prior work on programmer information needs and search behaviors, and evaluated these guidelines in a field study with over 300 participants who consulted instrumented documents over approximately seven months.

As part of the study environment, we used a novel dynamic format, named Casdoc, which we designed and presented in a prior publication [34]. Casdoc presents code examples with additional annotations that readers can selectively reveal and hide. Casdoc contrasts with the more common static layout of web-hosted documentation, reminiscent of printed books. The field study revealed that participants overwhelmingly favored Casdoc over a baseline format. The ability to only reveal the desired concise fragments avoided the common sources of distraction caused by additional content that a reader is not interested in. The study also elicited possible limitations that must be controlled. For example, selectively revealing information may cause readers to miss useful information if they do not know that they need it. Furthermore, small differences in the visual navigation hints can largely bias navigation behavior.

2 RELATED WORK

Prior research on documentation formats include the work of Curtis et al., who studied the symbology (e.g., natural language compared to ideograms) and spatial arrangement of documents [12], and of van der Meij et al., who reviewed the evolution of printed end-user tutorials [45]. More recently, researchers have investigated new media of documentation, such as video-based tutorials [29, 31, 47]. Past contributions also include guidelines and techniques to generate high-quality videos [11, 46] and code-based learning environments [35, 49]. Although they target relevant issues, these contributions do not address modern text-based online documents, which remain prevalent in the software documentation landscape.
Different techniques have also been proposed to improve the navigability of documents through various aids, for example by decomposing issue discussions based on an argumentation model or highlighting sentences in Stack Overflow threads to help programmers decide whether a post is relevant to their needs. These techniques can mitigate some issues related to an ineffective presentation of information, but they do not address the problem at its source. Closer to our work, Horvath et al. proposed a tool to create annotations on web pages. Similar to their work, we propose to augment documents with annotations, but we focus on annotations produced as part of the documentation authoring process.

Researchers and practitioners have also argued for and proposed techniques to add interactivity in online documents for the general public. For example, Victor argues that online documents should favor active reading by allowing readers to interact with the author’s arguments and claims. Holman et al. reviewed different techniques used to make documents such as news articles and research demonstrations interactive. The application context of, e.g., an average reader of a news article differs from that of a programmer trying to use an unfamiliar API for a specific task. However, better understanding interactivity features of general-purpose documents can help improve programmer-specific documents, and vice versa.

Past studies have elicited different types of information that programmers use in different contexts and their relation to different types of documents. Other researchers have focused on information seeking behaviors of programmers. Piroli and Card compared them to predators searching for prey, i.e., information, in documents. Others have elicited more nuanced personality profiles that vary among programmers. This line of work is crucial to understand the documentation needs of programmers, but more research is required to correctly translate their findings into actionable guidelines to design documents.

3 DOCUMENTATION PRESENTATION FEATURES UNDER STUDY

Documentation formats can vary across many dimensions. In this work, we studied text-based learning resources delivered through a web interface, with the goal of comparing a static and a dynamic presentation strategy. Within this theme, we focused on five concrete guidelines, each associated with a research question.

\[ \text{RQ1: Do programmers prefer an interactive format over a static one for software documents?} \]

\[ \text{RQ2[A-E]: What is the impact of [the guideline] on the navigation behaviors of the readers of a document?} \]

3.1 Presentation Guidelines

We synthesized the following five guidelines from prior work on programmer information needs and documentation reading behaviors.

A. Focus on Code Examples: The document format should emphasize high-quality code examples and help readers locate them. Tutorial authors recognized the importance of good code examples. Some tutorials are even accompanied by curated sets of standalone examples with minimal additional explanations. Code examples capture concrete partial solutions that programmers can copy and adapt to their needs. Hence, programmers commonly choose to first read the code examples in a document, and refer to the surrounding text only if some information is ambiguous. The absence of code examples is also generally considered a negative aspect of a document.

B. Reveal Information Gradually: The document format should reveal only some parts of the content at a time, ideally based on the needs of the reader. Being overly verbose and containing insufficient information are two common, yet conflicting issues of documentation. A document should contain enough information to satisfy the needs of a large audience. However, even if some information is present in a document, if a reader cannot find it quickly, they are likely to look for another document instead. Exposing readers to only parts of a document at a time can mitigate both issues. Collapsible HTML components can be used for that purpose, allowing readers to choose which information to reveal.

Tabbed containers can also be used to show information relevant for alternative technologies. The document thus caters to a larger audience while allowing readers to see only the information about the technology they prefer.

C. Split Information into Small Fragments: The document format should consist of concise, self-contained fragments. It is common for programmers to read a document out of sequence: they may look for a specific section related to their needs, skip information that they already know, or go back to an earlier point in the document to find background information about a concept. A set of concise and decoupled fragments can support such navigation behaviors. For example, the structure of API reference documentation consists of one fragment per API element, and it is generally not necessary to read one particular fragment to understand another. As an additional benefit, explicit fragments are easier to reuse to generate or augment other documents, e.g., for advanced query-answering systems.

D. Structure Information with Explicit Hints: The document format should include clear hints, separate from the main text, to guide readers as they navigate the structure of a document. Navigating within the content of a document is an important aspect of searching for information. Clear navigation links can help reduce this within-document navigation effort, so that readers find the information they seek before they start looking elsewhere. For example, traditional navigation aids, such as a table of contents, can remain visible and mark the reader’s position as they scroll down. Structural hints can provide a sense of location when readers navigate the document, particularly when they use small steps.
to locate the target information rather than trying to jump directly to it [42].

E. Support the Integration of External Content: The document format should provide a systematic way to integrate information from external sources. The extensive prior work on documentation generation and information retrieval (e.g., [26, 44]) constitutes a valuable opportunity to increase the coverage of a document. However, as the trustworthiness, authoritativeness, and tone of imported content can vary, it is important to clearly identify its source and to avoid inserting fragments that could degrade the quality of the original parts of a document. For example, Treude and Robillard implemented an approach to insert content from Stack Overflow in a floating box in the top right corner of reference documentation to avoid the risk of corrupting its original content [44].

3.2 Casdoc Documentation Format
In the field study, Casdoc served as an instrument to assess the impact of each guideline on the navigation behaviors of participants. In addition, Casdoc’s implementation also demonstrates the feasibility of implementing all guidelines in a single format and constitutes a concrete example of the application of each guideline. Our previous publication [34] describes the format in detail, and we summarize the most important aspects below.

Casdoc is an interactive format centered around code examples. The initial view of a Casdoc document consists of a high-quality top-level code example (Figure 1, left). Markers (blue underlines and gray brackets) indicate the presence of explanations, initially hidden, about some code elements. Readers reveal these explanations in floating annotations by hovering over the marker (Figure 1, middle). The annotation automatically disappears when the cursor leaves the marker or the annotation itself, allowing readers to reveal and hide annotations by moving the cursor over the code example.

Readers can also find markers inside an annotation, pointing to further nested annotations (Figure 1, right). Annotations are concise, as supporting details, e.g., a concept definition, are moved to nested annotations. Clicking on an annotation marker pins the annotation to the document. Readers can resize and move pinned annotations to personalize the presentation of information. To keep readers aware of the implicit graph structure of annotations, pinned nested annotations show a breadcrumb trail that readers can use to open a parent annotation. A search bar allows to locate information in any annotation, and two buttons allow to undo and redo the pinning actions.

Finally, Casdoc documents automatically import the reference documentation of standard library types and methods. This external content is inserted in their own annotations (Figure 1, middle), or in a separate part of a combined annotation if the marker for an original explanation would conflict with the anchor of reference documentation.

4 FIELD STUDY DESIGN
The field study took place during two consecutive sections of a third-year undergraduate course on software design with an important programming component. We provided part of the pedagogical material that students used during the course and analyzed their interaction with the documents to answer our research question. Each document we provided was available in two formats: a traditional, static format to use as a baseline, and the Casdoc format. Two different instructors, including one author of this paper, taught the two sections.

4.1 Ethical and Scientific Considerations
We designed the field study to maximize the ecological validity of the collected evidence and minimize the risks on participants. As members of the target audience for programmer documentation, students used the documents to satisfy genuine information needs. They accessed the documents through a public website, just like any other online resource.

We took great care not to pressure students into using a document format they were not comfortable with. We ensured that no personally identifiable data was collected and made this constraint explicit to students, so they would not fear that their participation in the study or their preference of document format may affect their performance in the course. Also, students did not have to participate in the study to access all documents in the baseline format on the same website, to avoid giving an unfair advantage in the course to participants. Finally, we designed the study to be minimally disruptive, to avoid unnecessary distractions for students and to limit cognitive biases related to being observed. Due to this constraint and to preserve the anonymity of participants, we excluded data collection methods such as surveys, interviews, and feedback forms.

These considerations encouraged participants to use the format they found the most valuable, even if it was the baseline format. Hence, we interpret a preference for one format as evidence of higher fitness for purpose of the format.

4.2 Document Creation
We prepared the corpus of documents for the study by inserting additional explanations in documents taken from the companion website of the course’s textbook. The companion website contains exercise statement, solution descriptions, and 126 code examples: 72 of them implement code described in the textbook (i.e., chapter code) and the other 54 show some solutions to the exercises (i.e., solution code). We only modified code examples, as exercise statements and solution descriptions were already sufficiently detailed.

For each code example, both formats contained the same additional explanations: the baseline format showed them as code comments, whereas Casdoc showed them as interactive annotations. Casdoc documents also contained the reference documentation of standard types and methods. The baseline format did not contain this external content to avoid extremely long code comments.

4.3 Data Collection
Participants accessed the documents on a public website. In the first course section, only the augmented code examples were hosted on the website. For the second section, we also hosted exercise statements and solution descriptions, without modification, to group all additional explanations: the baseline format showed them as code comments, whereas Casdoc showed them as interactive annotations. Casdoc documents also contained the reference documentation of standard types and methods. The baseline format did not contain this external content to avoid extremely long code comments.

We only collected data from students who provided informed consent. The study was approved by our institutional ethics review board.

https://github.com/prmr/DesignBook
Table 1: Events Collected During the Field Study

<table>
<thead>
<tr>
<th>Event</th>
<th>Origin</th>
<th>IDs</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit any page</td>
<td>server</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Consent to study</td>
<td>server</td>
<td>P/S</td>
<td></td>
</tr>
<tr>
<td>Withdraw consent</td>
<td>server</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Start new session</td>
<td>server</td>
<td>P/S</td>
<td></td>
</tr>
<tr>
<td>Open code example</td>
<td>server</td>
<td>P/S/D</td>
<td>format</td>
</tr>
<tr>
<td>Change format</td>
<td>server</td>
<td>P/S/D</td>
<td>new format</td>
</tr>
<tr>
<td>Open/Close annotation</td>
<td>client</td>
<td>P/S/D</td>
<td>annotation ID</td>
</tr>
<tr>
<td>Interact with marker</td>
<td>client</td>
<td>P/S/D</td>
<td>marker ID</td>
</tr>
<tr>
<td>Use search widget</td>
<td>client</td>
<td>P/S/D</td>
<td>query; selection(s)</td>
</tr>
<tr>
<td>Use navigation widgets</td>
<td>client</td>
<td>P/S/D</td>
<td>result</td>
</tr>
</tbody>
</table>

*This event was only collected during the second section of the course.

Documents in a single location and increase the adoption rate of the study’s website.

We instrumented the documents with asynchronous JavaScript functions to send events to an HTTP POST endpoint of the study’s website when participants interacted with Casdoc’s features. As the baseline format did not have any interactive feature, there was no event to log. However, the website also logged every document request by each participant in either format, allowing us to follow which document they consulted. For the second section of the course, we modified the website to additionally log all document requests, regardless of whether it was made by a participant but without capturing any information about the request origin. These additional events allowed us to assess the selection bias in our results.

To follow events performed by different participants, we stored three HTTP cookies in a participant’s browser. When a participant consented to the study, the website generated a random ID, sent in a persistent cookie that also served as proof that they agreed to participate in the study. The website also sent a second random ID in a session cookie. This cookie was reset when a participant closed and reopened their browser. Finally, a third persistent cookie retained the last format used by the participant, so that the next document would be opened in the same format. The value of this cookie was set by default to the Casdoc format when a participant consented to the study.

4.4 Collected Events

Table 1 summarizes the types of events we collected. The first six types of events are generated by the website, whereas the last four types are generated by JavaScript functions and sent through the HTTP POST endpoint. For each event, the website stored the type of event and a timestamp, as well as the related participant (P), session (S), or document (D) IDs and additional details as described in the last two columns of Table 1. Table 2 gives an overview of the data we collected. In total, 326 participants generated over 18,000 interaction events while consulting 7,338 code examples.

4.5 Limitations

HTTP cookies and AJAX requests are not as reliable as a controlled research environment, e.g., in a laboratory setting. A participant who cleared their cookies or changed web browsers during the study could be assigned multiple IDs. However, rigorously establishing the identity of participants would require the collection of personal information. To limit this threat, most of our analyses do not depend on the precise disambiguation of participants.

Although unlikely, it was also possible for a malicious individual to send artificial events to the POST endpoint. We mitigated this threat with a strategy based on a cryptographic function to detect tampering attempts, and by generating key events (e.g., opening a document) on the server side. We found no inconsistencies in the collected data.

Our participant sample is also susceptible to a selection bias. Students who participated in the study may be disproportionately enthusiastic about trying new technologies, whereas students satisfied with the baseline format did not need to be participants to use it. We assessed the magnitude of this bias by comparing the number of code example requests by participants (6,578) to all requests received by the website (14,644) during the second section.

These statistics show that our analysis ignores 55.1% of document requests. Some of these requests likely originate from students who
did not try Casdoc, but others may be due to web crawling bots. Nevertheless, the collected data captured the behavior of at least a considerable portion of our sampling frame.

Finally, as we only experimented with two formats, our results must be interpreted with care when applied to other formats. Students could have preferred a different baseline format over Casdoc, and some presentation guidelines that were beneficial to Casdoc could become detrimental if implemented differently or for a different type of documents. Further studies are required, using different formats and methodologies, to fully understand how the format of a document impacts its quality.

5 RESULTS

We reassembled the flat list of events into a meaningful structure to analyze our results. The actions of each participant are split into sessions, i.e., a period of continuous usage of the website. During a session, a participant viewed code example documents. Participants performed different actions on code examples, such as viewing an annotation and using the search widget. We considered all events performed within 15 minutes of consenting to the study as part of a learning period. We excluded the data of all participants who did not interact with the website beyond their learning period.

We observed that some participants left their browser open for many days or weeks, in which case the session ID cookie was never reset. Thus, we split long sessions whenever a participant did not generate any event for two consecutive hours. Within each session, opening a document initiates a new code example view, and all subsequent actions performed on this document are associated with this view. After artificially splitting long sessions, any document that remained opened initiates a new code example view in the second part of the session if the participant performed any action on the document. A single session can contain multiple views of the same document, if it is closed and reopened, and multiple views of different documents can overlap.

We grouped successive events associated with the same Casdoc annotation as a single annotation view action. Each annotation view starts with zero or more hovering events, optionally followed by a pin event, and a final optional unpin event. We grouped together multiple hovering events if they were less than five seconds apart, to account for participants accidentally moving outside the marker and immediately going back to it. To avoid spurious events, a hovering event was generated only if the participant hovered for at least one second over a marker.

As each keystroke in the search widget generated a new event, we grouped all events that incrementally built towards a single search query, as well as subsequent interactions with the search results, as a single search action. Each use of the breadcrumbs and the undo and redo buttons constitutes a separate action.

Table 3 describes the results of our field study. Only a minority of participants tried the baseline format at any point during the course, and the majority of them changed back to the Casdoc format shortly after. This suggests a positive answer to RQ1: participants generally preferred Casdoc over the static baseline.

As documents in both the baseline format and Casdoc were centered around code examples, we could not assess directly whether

<table>
<thead>
<tr>
<th>Table 3: Metrics and Results by Research Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ Metric</td>
</tr>
<tr>
<td>Participants</td>
</tr>
<tr>
<td>Sessions by all participants</td>
</tr>
<tr>
<td>Unique code examples</td>
</tr>
<tr>
<td>Code example views by all participants</td>
</tr>
<tr>
<td>Unique original annotations</td>
</tr>
</tbody>
</table>

For technical reasons, three code examples were not available during the first course section. We fixed this issue for the second section.

1 Preference for a Dynamic Format
Participants who used only Casdoc 49 129
Participants who tried the baseline format 5 21
... only during the learning phase 1 9
... for only one document 2 5
... for only one session (2+ documents) 1 1
... changed more than once 0 4
... kept baseline format until the end 1 5
Finding: Most participants only used Casdoc, and most of those who tried both formats changed back to Casdoc within the same session.

2A Focus on Code Examples
| Server-side code example requests | 11 268 |
| Annotation views | 356 | 1889 |
| Annotation view per code example view | 0.528 | 0.322 |
| % code example view with 1 annotation views | 17.8% | 14.2% |
| % original annotations never viewed | 42.2% | 60.9% |
| % participants who used annotations | 64.8% | 76.7% |
Finding: Most participants used annotations to find further information about elements of the code examples, but only for a minority of documents.

2B Reveal Information Gradually
| Annotation views | 356 | 1889 |
| Using only unpinned annotations | 311 | 1632 |
| Code example views with 1 annotation views | 129 | 832 |
| With 4+ annotation views | 93 | 696 |
| % original annotations that are nested | 18% | 9.4% |
Finding: Participants mostly viewed annotations in floating boxes, suggesting they can grasp the information quickly. Participants did not often have to combine information from many fragments to satisfy their needs.

2C Split Information into Small Fragments
| Annotation views | 356 | 1889 |
| Secondary navigation aid usage | 4 | 210 |
| ... search bar | 4 | 208 |
| ... undo/redo buttons | 0 | 1 |
| % code examples with navigation aid usage | 0.4% | 2.7% |
| Annotation views due to navigation aids | 0.6% | 1.4% |
| Annotation views due to code example marker | 187 | 1272 |
| ... blue underline under the anchor | 157 | 1039 |
| ... grey bracket marker in the left margin | 30 | 233 |
| % code example markers that are blue underlines | 63% | 63% |
Finding: Participants did not rely on secondary navigation aids, suggesting the markers are effective navigation hints. However, small differences in the visual appearance of markers impacted their effectiveness.

2D Structure Information with Explicit Hints
| Annotation views | 356 | 1889 |
| Secondary navigation aid usage | 4 | 210 |
| ... search bar | 4 | 208 |
| ... undo/redo buttons | 0 | 1 |
| % code examples with navigation aid usage | 0.4% | 2.7% |
Finding: API reference documentation augmented code examples with a considerable number of annotations. These imported annotations were used by participants, representing a quarter to over a third of all annotation views.
participants found this guideline beneficial. We observed that students looked at code examples—mainly chapter code—more often than exercise statements and solution descriptions. The number of solution code requests did not significantly increase or decrease as the course progressed, but was rather correlated to requests to solution descriptions (Kendall’s $\tau = 0.78$, $p = 0.0059$ [20]). As a preliminary answer to RQ2A, these observations suggest that documents focused on code example are useful to at least a considerable proportion of our participants.

Most participants used annotations to find further information about the code examples. They looked at a total of 356 and 1889 annotations during the first and second sections, respectively. Nevertheless, we observed that for most code example views, participants did not use annotations. This is not surprising, as the original code examples were designed to be self-contained by the textbook’s author. When participants looked at annotations, we found that they did not pin the annotation most of the time, and looked only rarely at nested annotations. Thus, we conclude for RQ2B and RQ2C that a gradual reveal of concise information fragments allows readers to grasp key information without being distracted by information they do not need, but it may also hide information that the reader does not know they need. To answer RQ2D, we interpret the low usage of secondary navigation aids, i.e., the search bar, undo and redo buttons, and breadcrumbs, as evidence that annotation markers clearly show the structure of each document, allowing participants to locate the information they needed. However, we also observed a disproportionately low usage of one type of markers: gray brackets found in the left margin of a document. This negative bias highlights the importance of seemingly small differences in visual cues.

Finally, we observed that importing reference documentation largely increased the number of annotations, which contributed to a considerable proportion of annotation views. Hence, to answer RQ2E, we see that integrating third-party content resulted in a tangible improvement of the documents for a minimal effort.

6 CONCLUSION
We conducted a field study to evaluate five presentation guidelines for web-hosted text-based software learning resources that we synthesized from prior work. The study used a novel presentation format, Casdoc, that dynamically reveals information based on the readers’ needs. During two sections of a course, we collected over 18,000 interaction events from 326 students accessing documents as part of their learning material. The data we collected suggest that interactive documents that selectively reveal only part of their content at a time are preferable to static documents. Our results also provide a better understanding of the benefits and limitations of the five guidelines we synthesized from prior work. For example, although visual cues can help readers understand the structure of a document, small graphical variations in the clues can have a large impact on their usefulness. This work will hopefully encourage researchers and technical writers to investigate in more detail the many possible design variations for presenting textual information, as part of the overall quality of a document.

ACKNOWLEDGMENTS
This work was funded by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES


