Title 3

ANNE-MARIE ROMMERDAHL, SDU, Denmark
JEREMY ALEXANDER RAMÍREZ GALEOTTI, SDU, Denmark
DIMITRIOS DAFNIS, SDU, Denmark
NASIFA AKTER, SDU, Denmark
MOHAMMAD HOSEIN KARDOUNI, SDU, Denmark
BEN TROVATO* and G.K.M. TOBIN*, Institute for Clarity in Documentation, USA
LARS THØRVÄLD, The Thørväld Group, Iceland
VALERIE BÉRANGER, Inria Paris-Rocquencourt, France

A clear and well-documented LaTeX document is presented as an article formatted for publication by ACM in a conference proceedings or journal publication. Based on the “acmart” document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

CCS Concepts: • **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Additional Key Words and Phrases: Do, Not, Use, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
Anne-Marie Rommerdahl, Jeremy Alexander Ramírez Galeotti, Dimitrios Dafnis, Nasifa Akter, Mohammad Hosein Kardouni, Ben Trovato, G.K.M. Tobin, Lars Thørväld, and Valerie Béranger. 2018. Title 3. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 15 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

ACM's consolidated article template, introduced in 2017, provides a consistent LaTeX style for use across ACM publications, and incorporates accessibility and metadata-extraction functionality necessary for future Digital Library endeavors. Numerous ACM and SIG-specific LaTeX templates have been examined, and their unique features incorporated into this single new template.

*Both authors contributed equally to this research.

Authors' Contact Information: Anne-Marie Rommerdahl, SDU, Odense, Denmark, anrom25@student.sdu.dk; Jeremy Alexander Ramírez Galeotti, SDU, Odense, Denmark, jeram25@student.sdu.dk; Dimitrios Dafnis, SDU, Odense, Denmark, didaf25@student.sdu.dk; Nasifa Akter, SDU, Copenhagen, Denmark, naakt23@student.sdu.dk; Mohammad Hosein Kardouni, SDU, Odense, Denmark, mokar25@student.sdu.dk; Ben Trovato, trovato@corporation.com; G.K.M. Tobin, webmaster@marysville-ohio.com, Institute for Clarity in Documentation, Dublin, Ohio, USA; Lars Thørväld, The Thørväld Group, Hekla, Iceland, larst@affiliation.org; Valerie Béranger, Inria Paris-Rocquencourt, Rocquencourt, France.

If you are new to publishing with ACM, this document is a valuable guide to the process of preparing your work for publication. If you have published with ACM before, this document provides insight and instruction into more recent changes to the article template.

The “`acmart`” document class can be used to prepare articles for any ACM publication — conference or journal, and for any stage of publication, from review to final “camera-ready” copy, to the author’s own version, with *very* few changes to the source.

2 Background and Related Work

Software reuse is a broad term, that refers to the practice of reusing previously written code, rather than coding from scratch. It is one of the key practices of software engineering. It is in fact such an important part of software engineering, that one of the ways to measure the quality of software is by it’s ’Reusability’[9] - i.e. the degree to which the application or its components can be reused. There are many different ways to do reuse in software engineering. Software libraries and frameworks are good examples of software that are intended to be reused. Developers may also scour the internet for things such as open-source software, or code snippets from websites like StackOverFlow, which can be reused.

There are multiple benefits to software reuse, depending on how the reuse is performed. One example is saving time. Not only can the developer avoid spending time writing the syntax of the code, they may also be able to avoid figuring out the logic of the software, and testing the reused software (assuming the software is tested by its creator). Another benefit is found through modularity. By breaking down a software system into smaller modules, the logic beind features or functions can be contained within a module, and can be tested thoroughly.

Despite reuse being an important practice in software engineering, there is still a limited focus on this practice when it comes to low-code development platforms (LCDP). This lack of reuse focus can easily impact the so-called ’Citizen Developers’, who have little or no coding knowledge, and may thus miss out on the benefits of reuse. A study from 2021 studied several low-code platforms (LCPs), in order to identify characteristic features of LCPs. The identified features were presented according to how frequent they occured, with domain-specific reference artifacts being categorized as ’rare’. Most studied systems offered catalogs of "reusable functions or examples of predefined processes", but they were found to be generic, or have a limited scope[10]. There have been proposed some ideas on how to promote reuse for LCPs, such as the strongly-typed rich templating language OSTRICH, developed for the model-driven low-code platform OutSystems. OutSystems provides scaffolding mechanisms for common development patterns and sample screen templates, both designed by experts on domain-specific languages (DSL). The practice of using templates in the OutSystems platform involves cloning and modifying samples, which may require more knowledge than the end-user posseses. The goal of OSTRICH is to remove this need for adapation when using templates, to remove the knowledge-barrier when making use of the available templates. This is done by abstracting and parameterizing the templates. A limitation of OSTRICH, is that it currently only supports the top nine most used production-ready screen templates from OutSystems. The end-user may not create and save their own templates, nor can they re-apply a template which they have customized.

Another approach focused on enabling model reuse by converting and merging heterogeneous models together into several graphs, which are then merged into one single graph (The Knowledge Graph), which acts as the repository of models. The Knowledge Graph can be queried to predict the next modeling step, based on the model being constructed by the user. This approach focuses on how to store, query, recommend and integrate the pre-defined models effeciently. End-Users can also persist their own models to the repository for later reuse.

For citizen developers, this feature of recommending models which have been constructed by domain experts and then

developed by model experts could prove very useful. However, while the user may persist their own models, the study is clearly not focused on guiding the user towards reusing their own models.
On the other hand, some existing LCDPs offer the user the ability to create their own models - for example by defining a new block in a block-based tool[39].

Building on the ideas discussed for improving reuse in low-code development platforms (LCDPs), several popular tools show these concepts in action. For instance, Webflow[32] is a leading low-code platform that offers a wealth of features for building responsive websites. One of its standout features is the ability to create reusable components and UI kits, which can significantly speed up the development process. With Webflow's intuitive interface, developers can quickly design and prototype components, and then reuse them across multiple pages and projects. Despite all of the useful features that this tools has, it does not provide guidance to the end-users to create custom reusable components which is the key feature of our project.

In a similar way, Mendix[40] takes this further for full enterprise apps by offering shareable building blocks like simple actions (microflows) and UI parts that anyone on a team can grab and use again without recoding. Through its Marketplace, a free online hub, you can download ready templates, connectors for tools like Salesforce, and basic setups that fit right into new projects, making everything faster and more uniform. This approach builds on the flexibility seen in platforms like Webflow, but adds strong team tools and AI suggestions to spot and create reusable pieces, empowering even beginners to build complex apps while keeping reuse simple and widespread.

OutSystems[41] further enhances the concept of reuse in low-code development platforms by emphasizing rapid application delivery through its robust set of features. Like Webflow and Mendix, OutSystems also provides a library of reusable components and templates that help developers complete projects faster. Its user-friendly visual development environment allows users to easily drag and drop elements while connecting with existing systems. OutSystems also supports teamwork with built-in version control and feedback features, making it easy for teams to share and improve reusable components. Additionally, the platform uses AI to suggest the best solutions and components for specific tasks, helping to streamline the development process. By encouraging reuse at both individual and team levels, OutSystems enables organizations to create scalable applications quickly while ensuring quality and consistency.

In order to analyze how block-based robotics environments address reuse area, 4 representative platforms were compared: mBlock, MakeCode, SPIKE LEGO, VEXcode GO and Open Roberta. The comparison focused on three main dimensions of reuse: structural reuse (through user-defined blocks or functions), social reuse (through sharing or remixing existing projects), and interoperable reuse (through import/export capabilities).

Table 1. Block Based Robotics Environments Reuse Support

Platform	Structural Reuse	Social Reuse	Interoperable Reuse	Reuse Support
VEXcode GO	X	X		Medium
mBlock	X	X	X	Medium
MakeCode	X	X	X	Medium
Spike Lego	X		X	Low
Open Roberta		X		Low

In this context, "reuse support" represents a scale that measures how effectively each platform facilitates reuse-related features. High reuse support indicates that users can easily create, share, and adapt existing components or projects.

Medium reuse support suggests that some reuse mechanisms are available but limited in scope or flexibility. Low reuse support implies that the platform provides only minimal or restricted features to promote reuse and improve user productivity.

As shown in Table 1, although these platforms include reusability features, they are quite limited, as none of them provide users with clear guidance on how to use these tools effectively, which restricts their ability to fully leverage them.

Research also indicates that block based programming environments should guide the end users towards good code organization as many may lack the necessary knowledge or may become stuck due to errors.[15] Although block based programming tools like Blockly were invented to teach programming to beginners by simple examples, Mayr-Dorn et al. mention that it is possible to express even large and highly complex real-world robot programs with the language concepts offered by these kind of block-based tools. [33]

Lin and Weintrop (2021) noted that most existing research on block-based programming focuses on supporting the transition to text-based languages rather than exploring how features within BBP environments [31]—such as abstraction or reuse—can enhance learning outcomes . In contrast, our work emphasizes guided abstraction, helping users understand and practice modular design directly within block-based environments.

Techapalokul and Tilevich (2019) proposed extending the Scratch programming environment with facilities for reusing individual custom blocks to promote procedural abstraction and improve code quality. They observed that while Scratch enables remixing of entire projects, it lacks mechanisms for reusing smaller, modular pieces of code. Their work suggests that supporting such fine-grained code reuse could enhance programmer productivity, creativity, and learning outcomes. Building on this idea, our project applies similar principles within the OpenRoberta environment by automating the detection of duplicate code segments and guiding users toward creating reusable custom blocks. Adler et al. (2021) introduced a search-based refactoring approach to improve the readability of Scratch programs by automatically applying small code transformations, such as simplifying control structures and splitting long scripts. Their findings demonstrated that automated refactoring can significantly enhance code quality and readability for novice programmers. Building upon this concept, our project applies similar principles in the OpenRoberta environment, focusing on detecting duplicate code segments and guiding users toward creating reusable custom blocks to promote modularity and abstraction.[3].

Existing block-based environments provide mechanisms for reuse, but lack intelligent support to help users recognize and apply reuse in practice.

To address this gap, our project introduces a guided reuse assistant within the Open Roberta Lab environment. The tool is designed to help users identify and apply reuse more easily while creating their robot programs. It works by automatically scanning a user's block-based program to detect repeated code segments that appear in different parts of the workspace. Once these duplicates are found, the system highlights them visually, drawing the user's attention to patterns that could be simplified.

When repeated blocks are detected, the assistant suggests creating a reusable custom block (function). It then helps the user generate this new block by identifying the small differences between the repeated parts—such as numbers, variables, or parameters—and turning these differences into inputs for the new block. After the user confirms, the system automatically replaces all the repeated sequences with calls to the newly created reusable block.

By combining ideas from procedural abstraction (organizing code into meaningful, reusable parts) and automated refactoring (improving code through intelligent transformations), our tool aims to make block-based programming more structured and efficient. It encourages users to build programs that are modular and easier to maintain, helps reduce unnecessary repetition, and supports learning by making the concept of reuse clear and hands-on.

In summary, our work bridges the gap between existing theoretical approaches to software reuse and their real-world application in block-based programming environments. Through this guided and semi-automated approach, we aim to make reuse visible, understandable, and practical for end-users working in Open Roberta.

3 Study Design

3.1 Problem Investigation

3.1.1 Problem Context and Motivation. End-user development (EUD) for collaborative robots (cobots) presents unique challenges, particularly for users without formal programming training. In domains such as chemistry laboratories, educational robotics, and industrial settings, end-users need to program robots to perform specific tasks but often lack the software engineering knowledge to write maintainable, well-structured code.

One critical challenge in EUD is code reuse. Users frequently create repetitive code because they struggle to recognize duplicate patterns, lack knowledge about abstraction mechanisms, or find existing tools too complex to use effectively. This problem manifests in several ways: programs become unnecessarily long and difficult to maintain, small changes require modifications in multiple locations increasing the risk of errors, and users miss opportunities to learn fundamental programming concepts such as modularity and abstraction.

In visual programming environments like Open Roberta Lab, don't provide assistance in identifying when code should be reused or how to extract repeated sequences into reusable components.

3.1.2 Stakeholder Analysis.

- **Chemistry Laboratory Personnel:** Chemists and lab technicians who use cobots for repetitive tasks such as sample preparation, dispensing, mixing, and quality control procedures. They possess deep domain expertise in chemistry but limited programming knowledge, often creating long, repetitive programs that become difficult to maintain when adapting experimental protocols. Their primary need is to quickly create and modify robot programs without becoming programming experts.

Table 2. Functional and Non-Functional Requirements

Type	ID	Description	Priority
Functional	FR1	Detect duplicate/similar block sequences	High
	FR2	Visually highlight detected duplications	High
	FR3	Suggest creation of reusable custom blocks	High
	FR4	Allow users to accept/reject suggestions	High
Non-Functional	NFR1	Seamless Open Roberta Lab integration	High
	NFR2	Intuitive interface for end users	High
	NFR3	No interference with existing workflow	High
	NFR4	Clear visual feedback during detection	High

3.1.3 Artifact Requirements.

3.2 Treatment Design

Our treatment focuses on developing a guided reuse assistant for the OpenRoberta Lab environment. The purpose of this tool is to help users recognize when parts of their robot programs can be reused, and to make it easier for them to create reusable custom blocks. By doing this, we aim to reduce repetitive code and help users learn important programming concepts such as modularity and abstraction.

3.2.1 Overview of the Tool. The guided reuse assistant is built as an extension inside Open Roberta Lab, which uses the Blockly framework. The assistant runs directly in the web browser and interacts with the user's block workspace. Its main job is to look through the user's program, find repeated sequences of blocks, and guide the user in turning them into reusable blocks.

The tool works in three main steps:

(1) **Detecting Repeated Code:** The assistant automatically scans the user's program and searches for parts that look the same or very similar. These are marked as potential duplicates.
(2) **Highlighting and Suggesting Reuse:** Once duplicates are found, the system highlights them in the workspace and shows a message suggesting that these sections could be made into a reusable block (function). This helps users see repetition they might not have noticed before.
(3) **Helping the User Create a New Block:** If the user agrees to the suggestion, the assistant opens a small guide to help them create the new block. It automatically detects any small differences between the repeated parts, such as numbers or variable names, and turns them into inputs (parameters) for the new block. When the block is created, repeated code is replaced by the new reusable block.

3.3 Treatment Validation

The treatment validation for this study adopts a mixed-methods evaluation approach to assess the effectiveness of the proposed features for guiding users in creating reusable custom blocks within the OpenRoberta environment. Participants will be recruited from local educational institutions, specifically chemistry students and teachers who frequently engage in laboratory work. A sufficient number of (x) participants will be selected to ensure a diverse range of experience levels with block-based programming. The experimental setup will take place in a controlled environment, where participants will be divided into two groups: one using the enhanced OpenRoberta platform with guided block creation features, and the other using the standard version without these enhancements. The procedure will begin with a pre-test to evaluate participants' prior understanding of modular programming concepts, followed by a series of tasks in which they will create reusable blocks from given code segments. Participants' interactions with the platform will be observed throughout the experiment. Data collection will include both quantitative measures, such as task completion time and accuracy in creating reusable blocks and qualitative feedback obtained through post-task interview. For the qualitative feedback, both groups will have to repeat the task, with the group that initially used the enhanced OpenRoberta platform now using the standard version, while the other group will use the enhanced version. The analysis will compare performance metrics between the two groups and apply thematic analysis to the qualitative data to identify user experiences and perceptions of the new features' usability and effectiveness. This comprehensive evaluation will provide a detailed understanding of how useful and effective is the block creation guidance feature to the end-users.

4 Modifications

Modifying the template — including but not limited to: adjusting margins, typeface sizes, line spacing, paragraph and list definitions, and the use of the `\vspace` command to manually adjust the vertical spacing between elements of your work — is not allowed.

Your document will be returned to you for revision if modifications are discovered.

5 Typefaces

The "`acmart`" document class requires the use of the "Libertine" typeface family. Your TeX installation should include this set of packages. Please do not substitute other typefaces. The "`lmodern`" and "`ltimes`" packages should not be used, as they will override the built-in typeface families.

6 Title Information

The title of your work should use capital letters appropriately - https://capitalizemytitle.com/ has useful rules for capitalization. Use the `title` command to define the title of your work. If your work has a subtitle, define it with the `subtitle` command. Do not insert line breaks in your title.

If your title is lengthy, you must define a short version to be used in the page headers, to prevent overlapping text. The `title` command has a "short title" parameter:

```
\title[short title]{full title}
```

7 Authors and Affiliations

Each author must be defined separately for accurate metadata identification. As an exception, multiple authors may share one affiliation. Authors' names should not be abbreviated; use full first names wherever possible. Include authors' e-mail addresses whenever possible.

Grouping authors' names or e-mail addresses, or providing an "e-mail alias," as shown below, is not acceptable:

```
\author{Brooke Aster, David Mehldau}
\email{dave,judy,steve@university.edu}
\email{firstname.lastname@phillips.org}
```

The `authornote` and `authornotemark` commands allow a note to apply to multiple authors — for example, if the first two authors of an article contributed equally to the work.

If your author list is lengthy, you must define a shortened version of the list of authors to be used in the page headers, to prevent overlapping text. The following command should be placed just after the last `\author{}` definition:

```
\renewcommand{\shortauthors}{McCartney, et al.}
```

Omitting this command will force the use of a concatenated list of all of the authors' names, which may result in overlapping text in the page headers.

The article template's documentation, available at https://www.acm.org/publications/proceedings-template, has a complete explanation of these commands and tips for their effective use.

Note that authors' addresses are mandatory for journal articles.

8 Rights Information

Authors of any work published by ACM will need to complete a rights form. Depending on the kind of work, and the rights management choice made by the author, this may be copyright transfer, permission, license, or an OA (open access) agreement.

Regardless of the rights management choice, the author will receive a copy of the completed rights form once it has been submitted. This form contains LaTeX commands that must be copied into the source document. When the document source is compiled, these commands and their parameters add formatted text to several areas of the final document:

- the "ACM Reference Format" text on the first page.
- the "rights management" text on the first page.
- the conference information in the page header(s).

Rights information is unique to the work; if you are preparing several works for an event, make sure to use the correct set of commands with each of the works.

The ACM Reference Format text is required for all articles over one page in length, and is optional for one-page articles (abstracts).

9 CCS Concepts and User-Defined Keywords

Two elements of the "acmart" document class provide powerful taxonomic tools for you to help readers find your work in an online search.

The ACM Computing Classification System — https://www.acm.org/publications/class-2012 — is a set of classifiers and concepts that describe the computing discipline. Authors can select entries from this classification system, via https://dl.acm.org/ccs/ccs.cfm, and generate the commands to be included in the LaTeX source.

User-defined keywords are a comma-separated list of words and phrases of the authors' choosing, providing a more flexible way of describing the research being presented.

CCS concepts and user-defined keywords are required for for all articles over two pages in length, and are optional for one- and two-page articles (or abstracts).

10 Sectioning Commands

Your work should use standard LaTeX sectioning commands: `\section`, `\subsection`, `\subsubsection`, `\paragraph`, and `\subparagraph`. The sectioning levels up to `\subsusection` should be numbered; do not remove the numbering from the commands.

Simulating a sectioning command by setting the first word or words of a paragraph in boldface or italicized text is **not allowed.**

Below are examples of sectioning commands.

10.1 Subsection

This is a subsection.

10.1.1 Subsubsection. This is a subsubsection.

Paragraph. This is a paragraph.

Table 3. Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
$	4 in 5	Used in business
Ψ^2_1	1 in 40,000	Unexplained usage

Table 4. Some Typical Commands

Command	A Number	Comments
`\author`	100	Author
`\table`	300	For tables
`\table*`	400	For wider tables

Subparagraph This is a subparagraph.

11 Tables

The "`acmart`" document class includes the "`booktabs`" package — https://ctan.org/pkg/booktabs — for preparing high-quality tables.

Table captions are placed *above* the table.

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material are found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 3 is included in the input file; compare the placement of the table here with the table in the printed output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 4 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed output of this document.

Always use midrule to separate table header rows from data rows, and use it only for this purpose. This enables assistive technologies to recognise table headers and support their users in navigating tables more easily.

12 Math Equations

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

12.1 Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin . . . \end` construction or with the short form `$ . . . $`. You can use any of the symbols and structures, from α to ω, available in LaTeX [29]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\rightarrow \infty}x=0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

12.2 Display Equations

A numbered display equation—one set off by vertical space from the text and centered horizontally—is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\rightarrow \infty}x=0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty}x_i=\int_{0}^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

13 Figures

The "`figure`" environment should be used for figures. One or more images can be placed within a figure. If your figure contains third-party material, you must clearly identify it as such, as shown in the example below.

Your figures should contain a caption which describes the figure to the reader.

Figure captions are placed *below* the figure.

Every figure should also have a figure description unless it is purely decorative. These descriptions convey what's in the image to someone who cannot see it. They are also used by search engine crawlers for indexing images, and when images cannot be loaded.

A figure description must be unformatted plain text less than 2000 characters long (including spaces). **Figure descriptions should not repeat the figure caption – their purpose is to capture important information that is not already provided in the caption or the main text of the paper.** For figures that convey important and complex new information, a short text description may not be adequate. More complex alternative descriptions can be placed in an appendix and referenced in a short figure description. For example, provide a data table capturing the information in a bar chart, or a structured list representing a graph. For additional information regarding how best to write figure descriptions and why doing this is so important, please see https://www.acm.org/publications/taps/describing-figures/.

13.1 The "Teaser Figure"

A "teaser figure" is an image, or set of images in one figure, that are placed after all author and affiliation information, and before the body of the article, spanning the page. If you wish to have such a figure in your article, place the command immediately before the `\maketitle` command:

Fig. 1. 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (https://goo.gl/VLCRBB).

```
\begin{teaserfigure}
  \includegraphics[width=\textwidth]{sampleteaser}
  \caption{figure caption}
  \Description{figure description}
\end{teaserfigure}
```

14 Citations and Bibliographies

The use of BibTEX for the preparation and formatting of one's references is strongly recommended. Authors' names should be complete — use full first names ("Donald E. Knuth") not initials ("D. E. Knuth") — and the salient identifying features of a reference should be included: title, year, volume, number, pages, article DOI, etc.

The bibliography is included in your source document with these two commands, placed just before the `\end{document}` command:

```
\bibliographystyle{ACM-Reference-Format}
\bibliography{bibfile}
```

where "`bibfile`" is the name, without the "`.bib`" suffix, of the BibTEX file.

Citations and references are numbered by default. A small number of ACM publications have citations and references formatted in the "author year" style; for these exceptions, please include this command in the **preamble** (before the command "`\begin{document}`") of your LATEX source:

```
\citestyle{acmauthoryear}
```

Some examples. A paginated journal article [2], an enumerated journal article [14], a reference to an entire issue [13], a monograph (whole book) [28], a monograph/whole book in a series (see 2a in spec. document) [22], a divisible-book such as an anthology or compilation [17] followed by the same example, however we only output the series if the volume number is given [18] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book [47], a chapter in a divisible book in a series [16], a multi-volume work as book [27], a couple of articles in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [4, 20], a proceedings article with all possible elements [46], an example of an enumerated proceedings article [19], an informally published work [21], a couple of preprints [7, 11], a doctoral dissertation [12], a master's thesis: [5], an online document / world wide web resource [1, 36, 48], a video game (Case 1) [35] and (Case 2) [34] and [30] and (Case 3) a patent [45], work accepted for publication [42], 'YYYYb'-test for prolific author [43] and [44]. Other cites might contain 'duplicate' DOI and URLs (some SIAM articles) [26]. Boris / Barbara Beeton: multi-volume works as books [24] and [23]. A presentation [38]. An article under review [8]. A couple of citations with DOIs: [25, 26]. Online citations: [48–50]. Artifacts: [37] and [6].

15 Acknowledgments

Identification of funding sources and other support, and thanks to individuals and groups that assisted in the research and the preparation of the work should be included in an acknowledgment section, which is placed just before the reference section in your document.

This section has a special environment:

```
\begin{acks}
...
\end{acks}
```

so that the information contained therein can be more easily collected during the article metadata extraction phase, and to ensure consistency in the spelling of the section heading.

Authors should not prepare this section as a numbered or unnumbered `\section`; please use the "`acks`" environment.

16 Appendices

If your work needs an appendix, add it before the "`\end{document}`" command at the conclusion of your source document.

Start the appendix with the "`appendix`" command:

```
\appendix
```

and note that in the appendix, sections are lettered, not numbered. This document has two appendices, demonstrating the section and subsection identification method.

17 Multi-language papers

Papers may be written in languages other than English or include titles, subtitles, keywords and abstracts in different languages (as a rule, a paper in a language other than English should include an English title and an English abstract). Use `language=...` for every language used in the paper. The last language indicated is the main language of the paper. For example, a French paper with additional titles and abstracts in English and German may start with the following command

```
\documentclass[sigconf, language=english, language=german,
               language=french]{acmart}
```

The title, subtitle, keywords and abstract will be typeset in the main language of the paper. The commands `\translatedXXX`, `XXX` begin title, subtitle and keywords, can be used to set these elements in the other languages. The environment `translatedabstract` is used to set the translation of the abstract. These commands and environment have a mandatory first argument: the language of the second argument. See `sample-sigconf-i13n.tex` file for examples of their usage.

18 SIGCHI Extended Abstracts

The "`sigchi-a`" template style (available only in LaTeX and not in Word) produces a landscape-orientation formatted article, with a wide left margin. Three environments are available for use with the "`sigchi-a`" template style, and produce formatted output in the margin:

`sidebar`: Place formatted text in the margin.
`marginfigure`: Place a figure in the margin.
`margintable`: Place a table in the margin.

Acknowledgments

To Robert, for the bagels and explaining CMYK and color spaces.

References

[1] Rafal Ablamowicz and Bertfried Fauser. 2007. *CLIFFORD: a Maple 11 Package for Clifford Algebra Computations, version 11.* Retrieved February 28, 2008 from http://math.tntech.edu/rafal/cliff11/index.html
[2] Patricia S. Abril and Robert Plant. 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan. 2007), 36–44. doi:10.1145/1188913.1188915
[3] Felix Adler, Gordon Fraser, Eva Gründinger, Nina Körber, Simon Labrenz, Jonas Lerchenberger, Stephan Lukasczyk, and Sebastian Schweikl. 2021. Improving Readability of Scratch Programs with Search-Based Refactoring. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-SEET)*. IEEE. doi:10.1109/ICSE-Companion.2021.00105
[4] Sten Andler. 1979. Predicate Path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226–236. doi:10.1145/567752.567774
[5] David A. Anisi. 2003. *Optimal Motion Control of a Ground Vehicle.* Master's thesis. Royal Institute of Technology (KTH), Stockholm, Sweden.
[6] Sam Anzaroot and Andrew McCallum. 2013. *UMass Citation Field Extraction Dataset.* Retrieved May 27, 2019 from http://www.iesl.cs.umass.edu/data/data-umasscitationfield
[7] Sam Anzaroot, Alexandre Passos, David Belanger, and Andrew McCallum. 2014. *Learning Soft Linear Constraints with Application to Citation Field Extraction.* arXiv:1403.1349 doi:10.48550/arXiv.1403.1349
[8] R. Baggett, M. Simecek, C. Chambellan, K. Tsui, and M. Fraune. 2025. Fluidity in the Phased Framework of Technology Acceptance: Case Study to Gain a Holistic Understanding of (Older Adult) Participant Advancement Through Acceptance Phases with Mobile Telepresence Robots. *Robotics Aut. Systems.* Manuscript submitted for review.
[9] Len Bass, Paul Clements, and Rick Kazman. 2021. *Software Architecture in Practice, 4th Edition.* Addison-Wesley Professional.
[10] Alexander Bock and Ulrich Frank. 2021. Low-Code Platform. *Business and Information Systems Engineering* 63 (2021). doi:10.1007/s12599-021-00726-8

[11] Lutz Bornmann, K. Brad Wray, and Robin Haunschild. 2019. *Citation concept analysis (CCA)—A new form of citation analysis revealing the usefulness of concepts for other researchers illustrated by two exemplary case studies including classic books by Thomas S. Kuhn and Karl R. Popper.* arXiv:1905.12410 [cs.DL]

[12] Kenneth L. Clarkson. 1985. *Algorithms for Closest-Point Problems (Computational Geometry).* Ph. D. Dissertation. Stanford University, Palo Alto, CA. UMI Order Number: AAT 8506171.

[13] Jacques Cohen (Ed.). 1996. Special issue: Digital Libraries. *Commun. ACM* 39, 11 (Nov. 1996).

[14] Sarah Cohen, Werner Nutt, and Yehoshua Sagic. 2007. Deciding equivalances among conjunctive aggregate queries. *J. ACM* 54, 2, Article 5 (April 2007), 50 pages. doi:10.1145/1219092.1219093

[15] Christian Gustavo Cossio-Mercado and Gonzalo Pablo Fernández. 2025. Challenges in the development of a block-based programming environment for Arduino. In *50a Conferencia Latinoamericana de Informática (CLEI).* https://www.researchgate.net/publication/396119595

[16] Bruce P. Douglass, David Harel, and Mark B. Trakhtenbrot. 1998. Statecarts in use: structured analysis and object-orientation. In *Lectures on Embedded Systems*, Grzegorz Rozenberg and Frits W. Vaandrager (Eds.). Lecture Notes in Computer Science, Vol. 1494. Springer-Verlag, London, 368–394. doi:10.1007/3-540-65193-4_29

[17] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago, Chapter The title of the chapter, 127–238. doi:10.1007/3-540-09237-4

[18] Ian Editor (Ed.). 2008. *The title of book two* (2nd. ed.). University of Chicago Press, Chicago, Chapter 100, 25–137. doi:10.1007/3-540-09237-4

[19] Matthew Van Gundy, Davide Balzarotti, and Giovanni Vigna. 2007. Catch me, if you can: Evading network signatures with web-based polymorphic worms. In *Proceedings of the first USENIX workshop on Offensive Technologies (WOOT '07).* USENIX Association, Berkley, CA, Article 7, 9 pages.

[20] Torben Hagerup, Kurt Mehlhorn, and J. Ian Munro. 1993. Maintaining Discrete Probability Distributions Optimally. In *Proceedings of the 20th International Colloquium on Automata, Languages and Programming (Lecture Notes in Computer Science, Vol. 700).* Springer-Verlag, Berlin, 253–264.

[21] David Harel. 1978. *LOGICS of Programs: AXIOMATICS and DESCRIPTIVE POWER.* MIT Research Lab Technical Report TR-200. Massachusetts Institute of Technology, Cambridge, MA.

[22] David Harel. 1979. *First-Order Dynamic Logic.* Lecture Notes in Computer Science, Vol. 68. Springer-Verlag, New York, NY. doi:10.1007/3-540-09237-4

[23] Lars Hörmander. 1985. *The analysis of linear partial differential operators. III.* Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Vol. 275. Springer-Verlag, Berlin, Germany. viii+525 pages. Pseudodifferential operators.

[24] Lars Hörmander. 1985. *The analysis of linear partial differential operators. IV.* Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Vol. 275. Springer-Verlag, Berlin, Germany. vii+352 pages. Fourier integral operators.

[25] IEEE 2004. IEEE TCSC Executive Committee. In *Proceedings of the IEEE International Conference on Web Services (ICWS '04).* IEEE Computer Society, Washington, DC, USA, 21–22. doi:10.1109/ICWS.2004.64

[26] Markus Kirschmer and John Voight. 2010. Algorithmic Enumeration of Ideal Classes for Quaternion Orders. *SIAM J. Comput.* 39, 5 (Jan. 2010), 1714–1747. doi:10.1137/080734467

[27] Donald E. Knuth. 1997. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms (3rd. ed.).* Addison Wesley Longman Publishing Co., Inc., Boston.

[28] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY.

[29] Leslie Lamport. 1986. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, MA.

[30] Newton Lee. 2005. Interview with Bill Kinder: January 13, 2005. Video. *Comput. Entertain.* 3, 1, Article 4 (Jan.-March 2005). doi:10.1145/1057270.1057278

[31] Yuhan Lin and David Weintrop. 2021. The Landscape of Block-Based Programming: Characteristics of Block-Based Environments and How They Support the Transition to Text-Based Programming. *Journal of Computer Languages* 67 (2021), 101075. doi:10.1016/j.cola.2021.101075

[32] Vlad Magdalin. 2012. Low code platform tool Webflow. https://webflow.com/.

[33] Christoph Mayr-Dorn, Mario Winterer, Christian Salomon, Doris Hohensinger, and Rudolf Ramler. 2021. Considerations for using Block-Based Languages for Industrial Robot Programming – a Case Study. In *Proceedings of the Conference on Industrial Robot Programming.* Johannes Kepler University, Linz, Austria. Supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry for Digital and Economic Affairs, and the Province of Upper Austria in the frame of the COMET center SCCH..

[34] Dave Novak. 2003. Solder man. Video. In *ACM SIGGRAPH 2003 Video Review on Animation theater Program: Part I - Vol. 145 (July 27–27, 2003).* ACM Press, New York, NY, 4. doi:10.945/woot07-S422 http://video.google.com/videoplay?docid=6528042696351994555

[35] Barack Obama. 2008. A more perfect union. Video. Retrieved March 21, 2008 from http://video.google.com/videoplay?docid=6528042696351994555

[36] Poker-Edge.Com. 2006. Stats and Analysis. Retrieved June 7, 2006 from http://www.poker-edge.com/stats.php

[37] R Core Team. 2019. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

[38] Brian J. Reiser. 2014. Designing coherent storylines aligned with NGSS for the K-12 classroom. Presentation at National Science Education Leadership Association Meeting, Boston, MA, USA. https://www.academia.edu/6884962/

[39] Mitchel Resnick, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52 (11 2009), 60–67. doi:10.1145/1592761.1592779

[40] Derek Roos. 2005. Low code platform tool Mendix. https://www.mendix.com/.

[41] Paulo Rosado. 2011. Low code platform tool Outsystems. https://www.outsystems.com/.

[42] Bernard Rous. 2008. The Enabling of Digital Libraries. *Digital Libraries* 12, 3, Article 5 (July 2008). To appear.

[43] Mehdi Saeedi, Morteza Saheb Zamani, and Mehdi Sedighi. 2010. A library-based synthesis methodology for reversible logic. *Microelectron. J.* 41, 4 (April 2010), 185–194.

[44] Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi, and Zahra Sasanian. 2010. Synthesis of Reversible Circuit Using Cycle-Based Approach. *J. Emerg. Technol. Comput. Syst.* 6, 4 (Dec. 2010), 12 pages.

[45] Joseph Scientist. 2009. The fountain of youth. Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.

[46] Stan W. Smith. 2010. An experiment in bibliographic mark-up: Parsing metadata for XML export. In *Proceedings of the 3rd. annual workshop on Librarians and Computers (LAC '10, Vol. 3)*, Reginald N. Smythe and Alexander Noble (Eds.). Paparazzi Press, Milan Italy, 422–431.

[47] Asad Z. Spector. 1990. Achieving application requirements. In *Distributed Systems* (2nd. ed.), Sape Mullender (Ed.). ACM Press, New York, NY, 19–33. doi:10.1145/90417.90738

[48] Harry Thornburg. 2001. *Introduction to Bayesian Statistics.* Retrieved March 2, 2005 from http://ccrma.stanford.edu/~jos/bayes/bayes.html, archived at [https://web.archive.org/web/20240505055615/https://ccrma.stanford.edu/~jos/bayes/bayes.html]

[49] TUG 2017. *Institutional members of the TeX Users Group.* Retrieved May 27, 2017 from http://wwtug.org/instmem.html

[50] Boris Veytsman. 2017. *acmart—Class for typesetting publications of ACM.* Retrieved May 27, 2017 from http://www.ctan.org/pkg/acmart

A Research Methods

A.1 Part One

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi malesuada, quam in pulvinar varius, metus nunc fermentum urna, id sollicitudin purus odio sit amet enim. Aliquam ullamcorper eu ipsum vel mollis. Curabitur quis dictum nisl. Phasellus vel semper risus, et lacinia dolor. Integer ultricies commodo sem nec semper.

A.2 Part Two

Etiam commodo feugiat nisl pulvinar pellentesque. Etiam auctor sodales ligula, non varius nibh pulvinar semper. Suspendisse nec lectus non ipsum convallis congue hendrerit vitae sapien. Donec at laoreet eros. Vivamus non purus placerat, scelerisque diam eu, cursus ante. Etiam aliquam tortor auctor efficitur mattis.

B Online Resources

Nam id fermentum dui. Suspendisse sagittis tortor a nulla mollis, in pulvinar ex pretium. Sed interdum orci quis metus euismod, et sagittis enim maximus. Vestibulum gravida massa ut felis suscipit congue. Quisque mattis elit a risus ultrices commodo venenatis eget dui. Etiam sagittis eleifend elementum.

Nam interdum magna at lectus dignissim, ac dignissim lorem rhoncus. Maecenas eu arcu ac neque placerat aliquam. Nunc pulvinar massa et mattis lacinia.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009