

1 Title 3

2 ANNE-MARIE ROMMERDAHL, SDU, Denmark

3 JEREMY ALEXANDER RAMÍREZ GALEOTTI, SDU, Denmark

4 DIMITRIOS DAFNIS, SDU, Denmark

5 NASIFA AKTER, SDU, Denmark

6 MOHAMMAD HOSEIN KARDOUNI, SDU, Denmark

7 A clear and well-documented L^AT_EX 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.

8 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.

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

10 ACM Reference Format:

11 Anne-Marie Rommerdahl, Jeremy Alexander Ramírez Galeotti, Dimitrios Dafnis, Nasifa Akter, and Mohammad Hosein Kardouni. 2018.
12 Title 3. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*.
13 ACM, New York, NY, USA, 15 pages. <https://doi.org/XXXXXX.XXXXXXX>

14 1 Introduction

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

19 2 Background and Related Work

20 Software reuse is a broad term, that refers to the practice of reusing previously written code, rather than coding from
21 scratch. It is such an important part of software engineering, that one of the ways to measure the quality of software is
22 by it’s ‘Reusability’^[9] - i.e. the degree to which the application or its components can be reused. There are multiple
23 benefits to practicing reuse in software engineering. One developer could save time by using another developer’s
24 reusable component, rather than coding their own. The developer avoids both the work of writing the syntax and
25 designing the logic of the component. The developer can design their own reusable components, keeping all the logic

26 Authors’ Contact Information: Anne-Marie Rommerdahl, SDU, Odense, Denmark, anrom25@student.sdu.dk; Jeremy Alexander Ramírez Galeotti, SDU,
27 Odense, Denmark, jeram25@student.sdu.dk; Dimitrios Dafnis, SDU, Odense, Denmark, didaf25@student.sdu.dk; Nasifa Akter, SDU, Copenhagen,
28 Denmark, naakt23@student.sdu.dk; Mohammad Hosein Kardouni, SDU, Odense, Denmark, mokar25@student.sdu.dk.

29 Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not
30 made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components
31 of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on
32 servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

33 © 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

34 Manuscript submitted to ACM

35 Manuscript submitted to ACM

in one place, which can then be tested thoroughly. However, 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).

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 occurred, 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]. This lack of focus on promoting reuse may impact the so-called 'Citizen Developers', who have little or no coding knowledge, and whom may then miss out on the benefits of reuse.

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[32]. 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 possesses. The goal of OSTRICH is to remove this need for adaptation 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 reuse of models, by converting and merging models into a single graph (the Knowledge Graph), which acts as a repository of models[24]. This graph is used to provide recommendations to the end-user, based on the model they're currently building. While this feature of recommending models (either constructed by domain experts and then developed by model experts, or made by the end-user themselves) could prove very useful, the study is clearly not focused on guiding the user towards reusing their own models.

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[33] 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 tool has, it does not provide guidance to the end-users to create custom reusable components.

In a similar way, Mendix[39] 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. This tool does offer guidance for the end-users to create custom reusable components through its AI suggestions, a lot of times these suggestions are not accurate enough (how do we know this??**).

OutSystems[40] 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

105 reusable components. Additionally, the platform uses AI to suggest the best solutions and components for specific
 106 tasks. By encouraging reuse at both individual and team levels, OutSystems enables organizations to create scalable
 107 applications quickly while ensuring quality and consistency. Similarly to the previous tool explained, the AI suggestions
 108 that this tool provides are not always accurate to successfully guide the end-user to create custom reusable components
 109 (again, how do we know this??).
 110

111 In order to analyze how block-based robotics environments address reuse area, 4 representative platforms were
 112 compared: mBlock, MakeCode, SPIKE LEGO, VEXcode GO and Open Roberta. The comparison focused on three main
 113 dimensions of reuse: structural reuse (through user-defined blocks or functions), social reuse (through sharing or
 114 remixing existing projects), and interoperable reuse (through import/export capabilities).
 115

117 Table 1. Block Based Robotics Environments Reuse Support
 118

119 Platform	120 Structural Reuse	121 Social Reuse	122 Interoperable Reuse	123 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

127
 128 In this context, “reuse support” represents a scale that measures how effectively each platform facilitates reuse-related
 129 features. High reuse support indicates that users can easily create, share, and adapt existing components or projects.
 130 Medium reuse support suggests that some reuse mechanisms are available but limited in scope or flexibility. Low reuse
 131 support implies that the platform provides only minimal or restricted features to promote reuse and improve user
 132 productivity.
 133

134 As shown in Table 1, although these platforms include reusability features, they are quite limited, as none of them
 135 provide users with clear guidance on how to use these tools effectively, which restricts their ability to fully leverage
 136 them.
 137

138 Lin and Weintrop (2021) noted that most existing research on block-based programming focuses on supporting
 139 the transition to text-based languages rather than exploring how features within BBP environments [31]—such as
 140 abstraction or reuse—can enhance learning outcomes. In contrast, our work emphasizes guided abstraction, helping
 141 users understand and practice modular design directly within block-based environments.
 142

143 Techapalokul and Tilevich (2019) proposed extending the Scratch programming environment with facilities for
 144 reusing individual custom blocks to promote procedural abstraction and improve code quality. They observed that
 145 while Scratch enables remixing of entire projects, it lacks mechanisms for reusing smaller, modular pieces of code. Their
 146 work suggests that supporting such fine-grained code reuse could enhance programmer productivity, creativity, and
 147 learning outcomes. Building on this idea, our project applies similar principles within the OpenRoberta environment
 148 by automating the detection of duplicate code segments and guiding users toward creating reusable custom blocks.
 149 Adler et al. (2021) introduced a search-based refactoring approach to improve the readability of Scratch programs by
 150 automatically applying small code transformations, such as simplifying control structures and splitting long scripts.
 151 Their findings demonstrated that automated refactoring can significantly enhance code quality and readability for
 152 novice programmers. Building upon this concept, our project applies similar principles in the OpenRoberta environment,
 153

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 in the workspace. The system visually highlights the found duplicates, drawing the user's attention to patterns that could be simplified.

The tool also offers the functionality to create the custom block for the end-user, 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. The tool automatically replaces all relevant duplicate sequences with the new custom 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.

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. In the domain of Chemistry, one of the most prevalent and important tasks is performing experiments in labs in order to test a hypothesis, or to aid in the understanding of how chemicals react. Robots can be used in chemistry labs to automate experiments with great effect, as many experiments involve steps that are repetitive, and susceptible to human error - such as a step being overlooked, instructions being misread, etc. Automation of menial tasks will leave the chemists with more time for other work, and also comes with the added bonus of chemists not having to handle dangerous chemicals.

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 and small changes require modifications in multiple locations, increasing the risk of errors. Several visual programming environments, like OpenRoberta Lab, don't provide assistance in identifying when code should be reused or how to extract repeated sequences into reusable components. As lab work in chemistry involves many repetitive tasks, these challenges can easily become an obstacle for the chemists, which may turn them away from using cobots, as the inconvenience outweighs the benefits.

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

209 to maintain when adapting experimental protocols. Their primary need is to quickly create and modify robot
 210 programs without becoming programming experts.
 211

212
 213
 214 Table 2. Functional and Non-Functional Requirements
 215

Type	ID	Description	Priority
Functional	FR1	Detect duplicate/similar block sequences	High
	FR2	Visually highlight detected duplicates	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

231
 232 3.1.3 *Artifact Requirements.* The artifact requirements can be seen in table 2.
 233

234 3.2 Treatment Design 235

236 Our treatment focuses on developing a guided reuse assistant for the OpenRoberta Lab environment. The purpose
 237 of this tool is to help users recognize which parts of their robot programs can be reused, and to make it easier for
 238 them to create reusable custom blocks. By doing this, we aim to reduce repetitive code and help users learn important
 239 programming concepts such as modularity and abstraction.
 240

241
 242 3.2.1 *Overview of the Tool.* The guided reuse assistant is built as an extension inside Open Roberta Lab, which uses the
 243 Blockly framework. The assistant runs directly in the web browser and interacts with the user's block workspace. Its
 244 main job is to look through the user's program, find repeated sequences of blocks, and guide the user in turning them
 245 into reusable blocks.
 246

247 The tool works in three main steps:
 248

- 249 (1) **Detecting Repeated Code:** The assistant automatically scans the user's program and searches for parts that
 250 look the same or very similar. These are marked as potential duplicates.
- 251 (2) **Highlighting and Suggesting Reuse:** Once duplicates are found, the system highlights them in the workspace
 252 and shows a message suggesting that these sections could be made into a reusable block (function). This helps
 253 users see repetition they might not have noticed before.
- 254 (3) **Helping the User Create a New Block:** If the user agrees to the suggestion, the assistant opens a small guide
 255 to help them create the new block. It automatically detects any small differences between the repeated parts,
 256 such as numbers or variable names, and turns them into inputs (parameters) for the new block. When the block
 257 is created, repeated code is replaced by the new reusable block.
 258

261 3.3 Treatment Validation

262 The treatment validation for this study adopts a mixed-methods evaluation approach to assess the effectiveness of
263 the proposed features for guiding users in creating custom reusable components (blocks) within the OpenRoberta
264 environment.

265 3.3.1 *Participant Recruitment.* A total of 10 participants will be selected to ensure a diverse range of experience levels
266 with block-based programming. Time constraints and resource availability have influenced the decision to limit the
267 number of participants. Participants will be recruited from a diverse pool of individuals affiliated with the University
268 of Southern Denmark and the broader chemistry community. This group of participants includes chemistry teachers,
269 professional chemical engineers, and students currently enrolled in chemistry-intensive curricula. To ensure relevant
270 practical expertise, the selection specifically targets those who frequently engage in laboratory environments. The
271 experimental sessions will be conducted across a range of environments to accommodate participant availability.
272 Physical sessions will take place within the chemistry laboratories at the University of Southern Denmark (SDU) as
273 well as a private residential setting. For remote participants, sessions will be administered virtually using Discord for
274 communication and AnyDesk for remote desktop control.

275 3.3.2 *Ethical Considerations and Sampling.* Prior to the commencement of the study, all participants are required to sign a
276 consent form acknowledging their voluntary participation and granting permission for screen recording and data usage.
277 It should be noted that this recruitment strategy constitutes *convenience sampling*. As such, they may not represent the
278 general population.

279 3.3.2 *Task Execution.* The participants will initially be given a short introduction to the OpenRoberta UI, as well
280 as the mujoco robot simulator. They will then perform one task which is described by a set of pre-defined steps to
281 perform. This task has been specifically designed to promote the reusability aspect. The task is focused on the domain
282 of chemistry, as it is modelled after a real lab experiment performed by chemistry students at SDU.

283 The participants will be instructed to program the robot to execute the following sequence of operations:

- 284 (1) Move the robot arm above mix cylinder
- 285 (2) Mix the chemistry ingredients
- 286 (3) Move the robot arm above the analysis pad
- 287 (4) Analyze the sample
- 288 (5) If the solution is analyzed (use if statement) then show a response message in the laptop's screen
- 289 (6) Place the following three objects into their corresponding slots in the chemistry equipment toolbox:
 - 290 • Methanol cylinder
 - 291 • Chloroform syringe
 - 292 • Toluene syringe
- 293 (7) Important notes for the participants:
 - 294 • After placing an object to its slot in the toolbox **wait 2 seconds** before you move to pick a new one.
 - 295 • After placing the **chloroform syringe** to its slot, **move the robot arm up by 10 cm** before you move to pick
296 the next chemistry object
 - 297 • Click the **play** button on the bottom right corner to start the simulation
 - 298 • Click the **reset** button on the bottom right corner to reset the scene of the robot simulator

313 Most optimal solution pre-defined by the researchers:

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

Instead of creating a long linear sequence of blocks (hard-coding the movement for all three objects), the most optimal solution utilizes a **Custom Reusable Component** to handle the repetitive action of placing an object to its corresponding slot inside the equipment toolbox. This approach not only reduces redundancy but also enhances code maintainability and readability, aligning with best practices in software development.

341

342

343

344

All the participants will try to complete the task using both the standard and the enhanced version of OpenRoberta. Half of the participants will begin using the enhanced version of OpenRoberta, while the other half will start with the standard version. Participants' interactions with the platform will be observed throughout the task. Guidance will be provided from the researchers to the participants throughout the task.

350

351 *3.3.3 Data Gathering and Analysis.* Data collection focuses on both quantitative performance and qualitative feedback from participants:

352

353

354

355

356

357

358

359

360

361

362

This comprehensive evaluation will provide a detailed understanding of how useful and effective is the block creation guidance feature to the end-users.

363

364

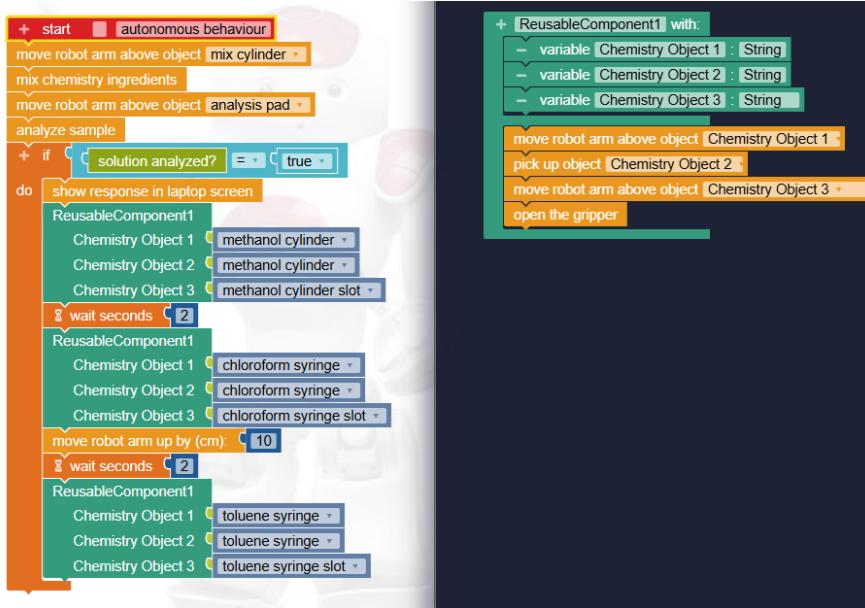


Fig. 1. The optimal solution implemented in OpenRoberta, utilizing a custom block for the object placement sequence.

365 4 Results

366 The treatment validation concluded with 10 participants total. The results show that all participants preferred the
 367 enhanced version of OpenRoberta Lab compared to the standard version, with 25% of participants finding the enhanced
 368 version to be 'better' than the original, and 75% found it to be 'much better'. [insert piechart]. Results also showed that
 369 75% of participants found the enhanced version 'easy' to use and 25% finding it 'very easy'.
 370

371 Results showed a high level of satisfaction with the highlights, with 87,5% of users being 'satisfied' or 'very satisfied',
 372 while 12,5% felt neutral about the highlights.
 373

374 5 Discussion

375 5.1 Lessons Learned

376 Based on the feedback from the participants, as well as observations of how they solved the task, the participants found
 377 the enhanced version of OpenRoberta Lab to be better than the standard version. Noteably, 7 out of 8 participants
 378 commented on how the enhanced version let them perform their task faster. As described in section 2, this is also one of
 379 the main benefits of reuse in the field of software engineering. While a somewhat large(?) percentage of the participants
 380 had no preference in regards to the visual look of the highlight, half of the users picked the 'Animated Color Highlight'.
 381 This suggests that dynamic visuals - in this case: the blocks changing color repeatedly - are well-suited for catching the
 382 user's attention.
 383

384 Changes suggested by the participants mainly focus on smaller customizations of the tool and the OpenRoberta
 385 Lab UI. It would be amiss to claim that the lack of suggested changes, focused on the tool overall, indicate that there
 386 is no need for improvement of the tool. As many of the participants consider themselves 'beginners' in regards to
 387 Computer Programming, it's likely that they lack ideas about other ways the tool could have been designed. Instead,
 388 these answers can be interpreted as the participants having little to no issue with the current design.
 389

390 5.2 Implications for Practice

391 5.3 Threats to Validity

392 5.3.1 *Convenience Sampling*. The participants to the study were either acquaintances of one of the authors of the study,
 393 or were recruited through these acquaintances. As such, the results of this study do not represent the general population
 394 within the domain of chemistry.
 395

396 5.3.2 *Limitations to observation*. Due to constraints with time and flexibility, only one of the authors was present to
 397 observe the participants. To ensure that data from the observation was not affected by this, a screen recording of each
 398 participant performing the task was saved. Several of the authors reviewed and discussed these recordings together to
 399 extract data.
 400

401 6 Tables

402 The "acmart" document class includes the "booktabs" package – <https://ctan.org/pkg/booktabs> – for preparing
 403 high-quality tables.
 404

405 Table captions are placed *above* the table.
 406

407 Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest
 408 their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's
 409 Manuscript submitted to ACM
 410

417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
Table 3. Frequency of Special Characters

Non-English or Math	Frequency	Comments
\emptyset	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	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

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 *L^AT_EX 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.

7 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.

7.1 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 L^AT_EX [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).

7.2 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 L^AT_EX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n \rightarrow \infty} x = 0 \quad (1)$$

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:

$$\sum_{i=0}^{\infty} x_i = \int_0^{\pi+2} f \quad (2)$$

just to demonstrate L^AT_EX's able handling of numbering.

8 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/>.

8.1 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:

```

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

```



Fig. 2. 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (<https://goo.gl/VLCRBB>).

9 Citations and Bibliographies

The use of Bib_TE_X 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 Bib_TE_X 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 L_AT_EX source:

573 \citetstyle{acmauthoryear}
 574
 575 Some examples. A paginated journal article [2], an enumerated journal article [14], a reference to an entire issue [13],
 576 a monograph (whole book) [28], a monograph/whole book in a series (see 2a in spec. document) [21], a divisible-book
 577 such as an anthology or compilation [16] followed by the same example, however we only output the series if the volume
 578 number is given [17] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book
 579 [46], a chapter in a divisible book in a series [15], a multi-volume work as book [27], a couple of articles in a proceedings
 580 (of a conference, symposium, workshop for example) (paginated proceedings article) [4, 19], a proceedings article with
 581 all possible elements [45], an example of an enumerated proceedings article [18], an informally published work [20], a
 582 couple of preprints [7, 11], a doctoral dissertation [12], a master's thesis: [5], an online document / world wide web
 583 resource [1, 36, 47], a video game (Case 1) [35] and (Case 2) [34] and [30] and (Case 3) a patent [44], work accepted for
 584 publication [41], 'YYYYb'-test for prolific author [42] and [43]. Other cites might contain 'duplicate' DOI and URLs
 585 (some SIAM articles) [26]. Boris / Barbara Beeton: multi-volume works as books [23] and [22]. A presentation [38]. An
 586 article under review [8]. A couple of citations with DOIs: [25, 26]. Online citations: [47–49]. Artifacts: [37] and [6].
 587
 588
 589

590 10 Acknowledgments

591
 592 Identification of funding sources and other support, and thanks to individuals and groups that assisted in the research
 593 and the preparation of the work should be included in an acknowledgment section, which is placed just before the
 594 reference section in your document.
 595

596 This section has a special environment:
 597

```

    598 \begin{acks}
    599 ...
    600 \end{acks}
  
```

601 so that the information contained therein can be more easily collected during the article metadata extraction phase, and
 602 to ensure consistency in the spelling of the section heading.
 603

604 Authors should not prepare this section as a numbered or unnumbered \section; please use the "acks" environment.
 605

606 11 Appendices

607
 608 If your work needs an appendix, add it before the "\end{document}" command at the conclusion of your source
 609 document.
 610

611 Start the appendix with the "appendix" command:
 612

```
\appendix
```

613 and note that in the appendix, sections are lettered, not numbered. This document has two appendices, demonstrating
 614 the section and subsection identification method.
 615

616 12 Multi-language papers

617
 618 Papers may be written in languages other than English or include titles, subtitles, keywords and abstracts in different
 619 languages (as a rule, a paper in a language other than English should include an English title and an English abstract).
 620 Use language=... for every language used in the paper. The last language indicated is the main language of the paper.
 621 For example, a French paper with additional titles and abstracts in English and German may start with the following
 622 command
 623

624 Manuscript submitted to ACM

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

627
 628 The title, subtitle, keywords and abstract will be typeset in the main language of the paper. The commands
 629 \translatedXXX, XXX begin title, subtitle and keywords, can be used to set these elements in the other languages. The
 630 environment `translatedabstract` is used to set the translation of the abstract. These commands and environment have
 631 a mandatory first argument: the language of the second argument. See `sample-sigconf-i13n.tex` file for examples of
 632 their usage.

633
 634

13 SIGCHI Extended Abstracts

635 The “sigchi-a” template style (available only in L^AT_EX and not in Word) produces a landscape-orientation formatted
 636 article, with a wide left margin. Three environments are available for use with the “sigchi-a” template style, and
 637 produce formatted output in the margin:

638 **sidebar**: Place formatted text in the margin.
 639 **marginfigure**: Place a figure in the margin.
 640 **maintable**: Place a table in the margin.

641
 642

Acknowledgments

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

644
 645

References

- 646 [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/rafacl/cliff11/index.html>
- 647 [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](https://doi.org/10.1145/1188913.1188915)
- 648 [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](https://doi.org/10.1109/ICSE-Companion.2021.00105)
- 649 [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](https://doi.org/10.1145/567752.567774)
- 650 [5] David A. Anisi. 2003. *Optimal Motion Control of a Ground Vehicle*. Master’s thesis. Royal Institute of Technology (KTH), Stockholm, Sweden.
- 651 [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>
- 652 [7] Sam Anzaroot, Alexandre Passos, David Belanger, and Andrew McCallum. 2014. *Learning Soft Linear Constraints with Application to Citation Field Extraction*. arXiv:[1403.1349](https://arxiv.org/abs/1403.1349) doi:[10.48550/arXiv.1403.1349](https://doi.org/10.48550/arXiv.1403.1349)
- 653 [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.
- 654 [9] Len Bass, Paul Clements, and Rick Kazman. 2021. *Software Architecture in Practice, 4th Edition*. Addison-Wesley Professional.
- 655 [10] Alexander Bock and Ulrich Frank. 2021. Low-Code Platform. *Business and Information Systems Engineering* 63 (2021). doi:[10.1007/s12599-021-00726-8](https://doi.org/10.1007/s12599-021-00726-8)
- 656 [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](https://arxiv.org/abs/1905.12410) [cs.DL]
- 657 [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.
- 658 [13] Jacques Cohen (Ed.). 1996. Special issue: Digital Libraries. *Commun. ACM* 39, 11 (Nov. 1996).
- 659 [14] Sarah Cohen, Werner Nutt, and Yehoshua Sagiv. 2007. Deciding equivalences among conjunctive aggregate queries. *J. ACM* 54, 2, Article 5 (April 2007), 50 pages. doi:[10.1145/1219092.1219093](https://doi.org/10.1145/1219092.1219093)

- [677] [15] Bruce P. Douglass, David Harel, and Mark B. Trakhtenbrot. 1998. Statecharts 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
- [678] [16] 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
- [679] [17] 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
- [680] [18] 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.
- [681] [19] 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.
- [682] [20] David Harel. 1978. *LOGICS of Programs: AXIOMATICs and DESCRIPTIVE POWER*. MIT Research Lab Technical Report TR-200. Massachusetts Institute of Technology, Cambridge, MA.
- [683] [21] 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
- [684] [22] 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.
- [685] [23] 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.
- [686] [24] Ilirian Ibrahimi and Dimitris Moudios. 2022. Towards model reuse in low-code development platforms based on knowledge graphs. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings* (Montreal, Quebec, Canada) (MODELS '22). Association for Computing Machinery, New York, NY, USA, 826–836. doi:10.1145/3550356.3561570
- [687] [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
- [688] [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
- [689] [27] Donald E. Knuth. 1997. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms* (3rd. ed.). Addison Wesley Longman Publishing Co., Inc., Boston.
- [690] [28] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY.
- [691] [29] Leslie Lamport. 1986. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, MA.
- [692] [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
- [693] [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.jcola.2021.101075
- [694] [32] Hugo Lourenço, Carla Ferreira, and João Costa Seco. 2021. OSTRICH - A Type-Safe Template Language for Low-Code Development. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 216–226. doi:10.1109/MODELS50736.2021.00030
- [695] [33] Vlad Magdalin. 2012. Low code platform tool Webflow. <https://webflow.com/>.
- [696] [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>
- [697] [35] Barack Obama. 2008. A more perfect union. Video. Retrieved March 21, 2008 from <http://video.google.com/videoplay?docid=6528042696351994555>
- [698] [36] Poker-Edge.Com. 2006. Stats and Analysis. Retrieved June 7, 2006 from <http://www.poker-edge.com/stats.php>
- [699] [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/>
- [700] [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/>
- [701] [39] Derek Roos. 2005. Low code platform tool Mendix. <https://www.mendix.com/>.
- [702] [40] Paulo Rosado. 2011. Low code platform tool Outsystems. <https://www.outsystems.com/>.
- [703] [41] Bernard Rous. 2008. The Enabling of Digital Libraries. *Digital Libraries* 12, 3, Article 5 (July 2008). To appear.
- [704] [42] 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.
- [705] [43] 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.
- [706] [44] Joseph Scientist. 2009. The fountain of youth. Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.
- [707] [45] 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.
- [708] [46] 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
- [709] [727] Manuscript submitted to ACM

- 729 [47] Harry Thorngrove. 2001. *Introduction to Bayesian Statistics*. Retrieved March 2, 2005 from <http://ccrma.stanford.edu/~jos/bayes/bayes.html>, archived
730 at [<https://web.archive.org/web/20240505055615/https://ccrma.stanford.edu/~jos/bayes/bayes.html>]
731 [48] TUG 2017. *Institutional members of the TeX Users Group*. Retrieved May 27, 2017 from <http://www.tug.org/instmem.html>
732 [49] Boris Veytsman. 2017. *acmart—Class for typesetting publications of ACM*. Retrieved May 27, 2017 from <http://www.ctan.org/pkg/acmart>

733
734 **A Research Methods**

735 **A.1 Part One**

736 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi malesuada, quam in pulvinar varius, metus nunc
737 fermentum urna, id sollicitudin purus odio sit amet enim. Aliquam ullamcorper eu ipsum vel mollis. Curabitur quis
738 dictum nisl. Phasellus vel semper risus, et lacinia dolor. Integer ultricies commodo sem nec semper.
739

740
741 **A.2 Part Two**

742 Etiam commodo feugiat nisl pulvinar pellentesque. Etiam auctor sodales ligula, non varius nibh pulvinar semper.
743 Suspendisse nec lectus non ipsum convallis congue hendrerit vitae sapien. Donec at laoreet eros. Vivamus non purus
744 placerat, scelerisque diam eu, cursus ante. Etiam aliquam tortor auctor efficitur mattis.
745

746
747 **B Online Resources**

748 Nam id fermentum dui. Suspendisse sagittis tortor a nulla mollis, in pulvinar ex pretium. Sed interdum orci quis metus
749 euismod, et sagittis enim maximus. Vestibulum gravida massa ut felis suscipit congue. Quisque mattis elit a risus ultrices
750 commodo venenatis eget dui. Etiam sagittis eleifend elementum.
751

752 Nam interdum magna at lectus dignissim, ac dignissim lorem rhoncus. Maecenas eu arcu ac neque placerat aliquam.
753 Nunc pulvinar massa et mattis lacinia.
754

755 Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009
756

757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780