Citation Report
List of articles citing

Towards a theory of the comprehension of computer prograr

DOI: 10.1016/s0020-7373(83)80031-5
International Journal of Man-Machine Studies, 1983, 18, 543-554.

Source: https://exaly.com/paper-pdf/16270719/citation-report.pdf
Version: 2024-04-28

This report has been generated based on the citations recorded by exaly.com for the above article. For the latest version of this publication list, visit the link given above.
The third column is the impact factor (IF) of the journal, and the fourth column is the number of citations of the article.

#	Paper	IF	Citations
458	Comprehension and recall of miniature programs. *International Journal of Man-Machine Studies*, **1984**, 21, 31-48		101
457	Beacons in computer program comprehension. *International Journal of Man-Machine Studies*, **1986**, 25, 697-709		63
456	Learning to program = learning to construct mechanisms and explanations. **1986**, 29, 850-858		311
455	Cap: A Knowledge Extraction Methodology for Computer Programming. **1986**, 30, 492-496		1
454	Eliciting knowledge for software development. **1987**, 6, 427-440		9
453	Cognitive processes in program comprehension. **1987**, 7, 325-339		99
452	Stimulus structures and mental representations in expert comprehension of computer programs. **1987**, 19, 295-341		300
451	Unscrambling non-sequential programs. **1988**, 18, 39-50		
450	.		3
449	GraphTrace□nderstanding object-oriented systems using concurrently animated views. **1988**, 23, 191-205		5
448	Automation, skills and the content of work. **1988**, 19, 1407-1418		1
447	Research on Computer Programming as a Cognitive Activity: implications for the study of classroom teaching. **1989**, 15, 177-189		2
446	Cognitive issues in the process of software development: review and reappraisal. *International Journal of Man-Machine Studies*, **1989**, 30, 171-191		9
445	Toward a theory of computer program bugs: an empirical test. *International Journal of Man-Machine Studies*, **1989**, 30, 23-46		33
444	The maintenance assistant: Work in progress. **1989**, 9, 3-17		6
443	Experimental evaluation of software documentation formats. **1989**, 9, 167-207		41
442	Hierarchical spiral model for information system and software development. Part 1: Theoretical background. **1990**, 32, 386-399		19

441	The impact of Pascal education on debugging skill. *International Journal of Man-Machine Studies*, **1990**, 33, 81-95	7
440	Variability in program design: the interaction of process with knowledge. *International Journal of Man-Machine Studies*, **1990**, 33, 305-322	13
439	An empirically-derived control structure for the process of program understanding. *International Journal of Man-Machine Studies*, **1990**, 33, 323-342	38
438	Typographic style is more than cosmetic. **1990**, 33, 506-520	45
437	. **1990**, 7, 39-45	18
436	The Role of Notation and Knowledge Representation in the Determination of Programming Strategy: A Framework for Integrating Models of Programming Behavior. **1991**, 15, 547-572	19
435	.	
434	What do novices learn during program comprehension?. **1991**, 3, 199-222	72
433	The System Design Process. **1991**, 267-340	
432	Instruction for software engineering expertise. **1991**, 271-282	2
431	Approaches to program comprehension. **1991**, 14, 79-84	24
430	The software maintenance of large software systems: Management, methods and tools. **1991**, 32, 135-154	6
429	The initial stage of program comprehension. *International Journal of Man-Machine Studies*, **1991**, 35, 517-540	23
428	Information relationships in PROLOG programs: how do programmers comprehend functionality?. *International Journal of Man-Machine Studies*, **1991**, 35, 313-328	27
427	Towards a model of programmers' cognitive processes in software maintenance: A structural learning theory approach for debugging. **1991**, 3, 85-106	5
426	Knowledge Creation and Retrieval in Program Design: A Comparison of Novice and intermediate Student Programmers. **1991**, 6, 1-46	69
425	.	8
424	.	1

423	.	
422	.	1
421	Intelligent fault localization in software. **1992**,	
420	Intelligent search and acquisition of business knowledge from programs. **1992**, 4, 1-17	18
419	A framework for software maintenance: A foundation for scientific inquiry. **1992**, 4, 105-117	4
418	On the re-engineering of transaction systems. **1992**, 4, 143-162	7
417	VPCL: A visual language for teaching and learning programming. (A picture is worth a thousand words). **1992**, 3, 299-317	6
416	The role of program structure in software maintenance. *International Journal of Man-Machine Studies*, **1992**, 36, 21-63	23
415	Understanding someone else's code: Analysis of experiences. **1993**, 23, 269-275	51
414	.	9
413	.	
412	.	4
411	.	35
410	.	7
409	.	9
408	A blackboard architecture for intelligent assistance in software maintenance.	
407	From code understanding needs to reverse engineering tool capabilities.	27
406	The psychology of computer languages for introductory programming courses. **1993**, 11, 213-228	6

405	.		2
404	.		
403	.		1
402	. **1994**,		6
401	A memory-based approach to recognizing programming plans. **1994**, 37, 84-93		61
400	. **1994**,		7
399	. *IEEE Transactions on Software Engineering*, **1994**, 20, 463-475	3.5	92
398	. *IEEE Transactions on Software Engineering*, **1994**, 20, 445-462	3.5	7
397	.		
396	.		
395	.		19
394	Maintenance and Evolution of Software Products. **1994**, 39, 1-49		0
393	Planning for Software Maintenance Education Within a Computee Science Framework. **1994**, 5, 1-13		2
392	Helping programmers understand computer programs. **1995**, 26, 25-46		13
391	. **1995**, 28, 44-55		258
390	Program Structure and Design. **1995**, 19, 507-562		57
389	A Knowledge-Based Approach to Program Understanding. **1995**,		2
388	.		8

387	Program Understanding: Models and Experiments. **1995**, 40, 1-38		18
386	Design of a generic reverse engineering assistant tool.		8
385	Detecting interleaving.		7
384	VIFOR 2: a tool for browsing and documentation. **1996**,		3
383	A query algebra for program databases. *IEEE Transactions on Software Engineering*, **1996**, 22, 202-217	3.5	30
382	Understanding Interleaved Code. **1996**, 47-76		
381	On the role of hypotheses during opportunistic understanding while porting large scale code.		11
380	A workbench for program comprehension during software maintenance.		8
379	Recursion vs. iteration: An empirical study of comprehension. **1996**, 32, 73-82		11
378	A method for documenting code components. **1996**, 34, 89-104		3
377	Cognitive processes in program comprehension: An empirical analysis in the Context of software reengineering. **1996**, 34, 177-189		18
376	Understanding interleaved code. **1996**, 3, 47-76		11
375	Programming pedagogy: a psychological overview. **1996**, 28, 17-22		252
374	The Gadfly: an approach to architectural-level system comprehension.		1
373	Towards a framework for program understanding.		33
372	Greater understanding through maintainer driven traceability.		3
371	Softwarewartung und Reengineering. **1996**,		
370	Responses to comprehension questions and verbal protocols as measures of computer program comprehension processes. **1997**, 16, 320-336		4

369	PUI: a tool to support program understanding.	2
368	On integrating visualization techniques for effective software exploration.	9
367	An empirical study of novice program comprehension in the imperative and object-oriented styles. **1997**,	29
366	.	2
365	Hypothesis-driven understanding processes during corrective maintenance of large scale software.	10
364	How do program understanding tools affect how programmers understand programs?.	41
363	Building a research infrastructure for program comprehension observations.	1
362	Incremental redocumentation with hypertext.	4
361	Towards standard for experiments in program comprehension.	10
360	Towards a precise description of reverse engineering methods and tools.	1
359	Cognitive design elements to support the construction of a mental model during software visualization.	36
358	Interactive Explanation of Software Systems. **1997**, 4, 53-75	8
357	Model-based design of reverse engineering tools. **1998**, 10, 353-380	3
356	Conditioned program slicing. **1998**, 40, 595-607	119
355	.	11
354	Developing the designer’s toolkit with software comprehension models.	2
353	Intent specifications: an approach to building human-centered specifications.	3
352	Task oriented software understanding.	7

351	FEPSS: a flexible and extensible program comprehension support system.	
350	The case for user-centered CASE tools. **1998**, 41, 93-99	50
349	Evaluating software maintenance support tools for their support of program comprehension.	2
348	Archetypal source code searches: a survey of software developers and maintainers.	30
347	Program understanding behavior during adaptation of large scale software.	15
346	The Relevance of Application Domain Knowledge: Characterizing the Computer Program Comprehension Process. **1998**, 15, 51-78	18
345	·	14
344	Task orientation and tailoring of interactive software explanations.	1
343	Cognitive design elements to support the construction of a mental model during software exploration. **1999**, 44, 171-185	131
342	Portability by automatic translation: A large-scale case study. **1999**, 107, 1-28	5
341	Episodic Indexing: A Model of Memory for Attention Events. **1999**, 23, 117-156	11
340	Mental representations of expert procedural and object-oriented programmers in a software maintenance task. **1999**, 50, 61-83	32
339	Introduction to the Special Issue Best of Empirical Studies of Programmers 7 **1999**, 51, 3-5	1
338	Program understanding behavior during corrective maintenance of large-scale software. **1999**, 51, 31-70	38
337	Novice comprehension of small programs written in the procedural and object-oriented styles. **1999**, 51, 71-87	62
336	Editorial: 30th Anniversary Issue. **1999**, 51, 119-124	
335	An evaluation of the cognitive processes of programmers engaged in software debugging. **1999**, 11, 73-91	5
334	·	23

333	A framework for analysing the effect of 'change' in legacy code. **1999**,		2
332	Browsing and searching software architectures. **1999**,		
331	Empirical evaluation of hypertextual information access from program text.		1
330	Building documentation generators. **1999**,		26
329	????????????????. **1999**, 16, 617-627		
328	A coding scheme to support systematic analysis of software comprehension. *IEEE Transactions on Software Engineering*, **1999**, 25, 526-540	3.5	16
327	Detecting the error threshold for rule-based programs: a logit model. *Expert Systems With Applications*, **2000**, 19, 229-233	7.8	
326	How do program understanding tools affect how programmers understand programs?. **2000**, 36, 183-207		43
325	An empirical analysis of debugging performance □differences between iterative and recursive constructs. **2000**, 54, 17-28		8
324	The role of comprehension in software inspection. **2000**, 52, 121-129		24
323	The use of domain knowledge in program understanding. **2000**, 9, 143-192		27
322	The Multimedia Maintenance Interface (MuMMI) system.		
321	Intent specifications: an approach to building human-centered specifications. *IEEE Transactions on Software Engineering*, **2000**, 26, 15-35	3.5	114
320	.		19
319	Direction and scope of comprehension-related activities by procedural and object-oriented programmers: an empirical study.		5
318	Inference-based and expectation-based processing in program comprehension.		10
317	Comparing two spreadsheet calculation paradigms: an empirical study with novice users. **2001**, 13, 427-446		2
316	Near-term memory in programming: a simulation-based analysis. **2001**, 54, 189-210		16

315	Focal structures and information types in Prolog. **2001**, 54, 211-236		5
314	An exploratory study of program comprehension strategies of procedural and object-oriented programmers. **2001**, 54, 1-23		28
313	Managing crosscutting concerns during software evolution tasks. **2002**,		13
312	Software Evolution and the Staged Model of the Software Lifecycle. **2002**, 1-54		5
311	Constructivism and program comprehension strategies.		3
310	.		0
309	Traceability recovery in RAD software systems.		13
308	Program comprehension by visualization in contexts.		
307	A model for understanding software components.		10
306	Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. *IEEE Transactions on Software Engineering*, **2002**, 28, 595-606	3.5	91
305	. *IEEE Transactions on Software Engineering*, **2002**, 28, 970-983	3.5	608
304	Theory-based analysis of cognitive support in software comprehension tools.		8
303	Experimental evaluation of hypertext access structures. **2002**, 14, 83-108		6
302	A semantic entropy metric. **2002**, 14, 293-310		14
301	A relational approach to defining and implementing transformations between metamodels. **2003**, 2, 215-239		35
300	A comparison of methods for locating features in legacy software. **2003**, 65, 105-114		57
299	Code and data spatial complexity: two important software understandability measures. **2003**, 45, 539-546		28
298	Developing relational navigation to effectively understand software.		2

297	Case study: reconnaissance techniques to support feature location using RECON2.	1
296	Learning and Teaching Programming: A Review and Discussion. **2003**, 13, 137-172	701
295	Applying the signature concept to plan-based program understanding.	1
294	Analogy of incremental program development and constructivist learning.	3
293	Exploring software systems.	6
292	A framework for understanding conceptual changes in evolving source code.	4
291	Programming at runtime: Requirements & paradigms for nonprogrammer web application development. **2003**,	2
290	Cognitive and social aspects of software engineering. **2003**, 35, 3-6	
289	Cognitive and social aspects of software engineering. **2003**,	1
288	Visualizing model mappings in UML. **2003**,	19
287	Verification of the cryptlib Kernel. **2004**, 167-213	
286	An initial approach to assessing program comprehensibility using spatial complexity, number of concepts and typographical style.	2
285	Comprehension Strategies of End-User Programmers in an Event-Driven Application.	6
284	A multi-national study of reading and tracing skills in novice programmers. **2004**,	63
283	MFV-class: a multi-faceted visualization tool of object classes. **2004**, 5, 1374-81	0
282	Expectation-based, inference-based, and bottom-up software comprehension. **2004**, 16, 427-447	10
281	Program comprehension and authentic measurement:. **2004**, 61, 169-185	11
280	Measurement of object-oriented software spatial complexity. **2004**, 46, 689-699	14

279	Combined software and hardware comprehension in reverse engineering.	1
278	Using feature modeling for program comprehension and software architecture recovery.	9
277	Program comprehension for Web services.	6
276	Structural Knowledge and Language Notational Properties in Program Comprehension.	0
275	Programming style changes in evolving source code.	1
274	Cognitive process during program debugging. **2004**,	2
273	Semantic driven program analysis.	6
272	A multi-national study of reading and tracing skills in novice programmers. **2004**, 36, 119-150	201
271	Towards understanding programs through wear-based filtering. **2005**,	56
270	An empirical study of programmer learning during incremental software development. **2005**,	
269	An investigation into professional programmers' mental representations of variables.	1
268	. **2005**,	11
267	Managing software change tasks: an exploratory study.	23
266	Presenting micro-theories of program comprehension in pattern form.	
265	A cognitive model for program comprehension. **2005**,	
264	Theories, methods and tools in program comprehension: past, present and future. **2005**,	72
263	Empirically studying software practitioners - bridging the gap between theory and practice. **2005**,	6
262	A First Look at Novice Compilation Behaviour Using BlueJ. **2005**, 15, 25-40	68

261	An Ontology-Based Approach to Software Comprehension - Reasoning about Security Concerns. **2006**,		12
260	A Context-Driven Software Comprehension Process Model. **2006**,		5
259	A Program Beacon Recognition Tool. **2006**,		2
258	Legacysoftware. **2006**,		
257	Using Sex Differences to Link Spatial Cognition and Program Comprehension. **2006**,		27
256	A Context-Aware Analysis Scheme for Bloom's Taxonomy.		8
255	Reverse Engineering in Support of Litigation: Experiences in an Adversarial Environment. **2006**,		
254	A study of design characteristics in evolving software using stability as a criterion. *IEEE Transactions on Software Engineering*, **2006**, 32, 315-329	3.5	44
253	The SEXTANT Software Exploration Tool. *IEEE Transactions on Software Engineering*, **2006**, 32, 753-768	3.5	9
252	Measurement of Object-Oriented Software Understandability Using Spatial Complexity. **2006**, 155-181		
251	a(MDA)2: A Model Driven Approach to Multi-Dimensional Separation of Concerns with OCL. **2006**, 163, 19-29		
250	Theories, tools and research methods in program comprehension: past, present and future. **2006**, 14, 187-208		61
249	A formal methods approach to medical device review. **2006**, 39, 61-67		42
248	An eye-tracking methodology for characterizing program comprehension processes. **2006**,		107
247	Transparency, holophrasting, and automatic layout applied to control structures for visual dataflow programming languages. **2006**,		
246	Using Abstraction-driven Slicing for Postmortem Analysis of Software.		1
245	Towards Spatial Complexity Measures for Comprehension of Java Programs. **2006**,		2
244	Supporting CS1 with a program beacon recognition tool. **2007**,		1

243	Spatial skills and navigation of source code. **2007**,		7
242	A Review of Australasian Investigations into Problem Solving and the Novice Programmer. **2007**, 17, 201-213		14
241	. **2007**,		11
240	Towards A Process-Oriented Software Architecture Reconstruction Taxonomy. **2007**,		30
239	Spatial skills and navigation of source code. **2007**, 39, 231-235		8
238	A Systematic Review of Theory Use in Software Engineering Experiments. *IEEE Transactions on Software Engineering*, **2007**, 33, 87-107	3.5	118
237	Software Visualization - A Process Perspective. **2007**,		1
236	Toward Reducing Fault Fix Time: Understanding Developer Behavior for the Design of Automated Fault Detection Tools. **2007**,		17
235	Source Code Analysis: A Road Map. **2007**,		80
234	Constructivist Learning During Software Development. **2007**, 1, 78-101		3
233	Debugging strategies and tactics in a multi-representation software environment. **2007**, 65, 992-1009		22
232	Comprehension strategies and difficulties in maintaining object-oriented systems: An explorative study. **2007**, 80, 1541-1559		20
231	An objective-oriented approach to program comprehension using multiple information sources. **2008**, 51, 825-847		
230	The impacts of function extraction technology on program comprehension: A controlled experiment. **2008**, 50, 1165-1179		7
229	Story-driven approach to software evolution. **2008**, 2, 304		7
228	Patterns for understanding frameworks. **2008**,		3
227	Don□ do this □Pitfalls in using anti-patterns in teaching human□omputer interaction principles. **2008**, 50, 979-1008		24
226	Asking and Answering Questions during a Programming Change Task. *IEEE Transactions on Software Engineering*, **2008**, 34, 434-451	3.5	172

225	Evaluating Key Statements Analysis. **2008**,		2
224	Expressiveness and effectiveness of program comprehension: Thoughts on future research directions. **2008**,		5
223	BEYOND INFORMATION SILOS AN OMNIPRESENT APPROACH TO SOFTWARE EVOLUTION. **2008**, 02, 431-468		11
222	Checklist Inspections and Modifications: Applying Bloom's Taxonomy to Categorise Developer Comprehension. **2008**,		2
221	Remixing visualization to support collaboration in software maintenance. **2008**,		4
220	Beyond generated software documentation A web 2.0 perspective. **2009**,		0
219	Non-programmers identifying functionality in unfamiliar code: Strategies and barriers. **2009**,		2
218	Mapping a sequence diagram to the related code: Cognitive levels expressed by developers. **2009**,		
217	Program Comprehension for User-Assisted Test Oracle Generation. **2009**,		1
216	IT Revolutions. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, **2009**,	0.2	0
215	. *IEEE Transactions on Software Engineering*, **2009**, 35, 305-324	3.5	4
214	. *IEEE Transactions on Software Engineering*, **2009**, 35, 573-591	3.5	178
213	Non-programmers identifying functionality in unfamiliar code: strategies and barriers. **2010**, 21, 263-276		26
212	An introduction to program comprehension for computer science educators. **2010**,		34
211	Influence of Synchronized Domain Visualizations on Program Comprehension. **2010**,		
210	An evaluation of object oriented example programs in introductory programming textbooks. **2010**, 41, 126-143		10
209	The Monitoring System for Electric Quantity Consumed in Extruder Based on WB Electrical Transducer. **2010**,		
208	Reactive information foraging for evolving goals. **2010**,		25

207	An eye tracking study on the effects of layout in understanding the role of design patterns. **2010**,		39
206	Estimating the Optimal Number of Latent Concepts in Source Code Analysis. **2010**,		26
205	Trustrace: Improving Automated Trace Retrieval through Resource Trust Analysis. **2011**,		2
204	Trust-Based Requirements Traceability. **2011**,		18
203	Automatic Segmentation of Method Code into Meaningful Blocks to Improve Readability. **2011**,		18
202	Advances in Artificial Intelligence. *Lecture Notes in Computer Science*, **2011**,	0.9	1
201	Empirical assessment of UML class diagram layouts based on architectural importance. **2011**,		8
200	Context and Vision: Studying Two Factors Impacting Program Comprehension. **2011**,		1
199	On the Quality of Examples in Introductory Java Textbooks. *ACM Transactions on Computing Education*, **2011**, 11, 1-21	2.1	13
198	Categorization of concerns. **2011**,		2
197	On the role of human thought. **2011**,		1
196	The Influence of the Task on Programmer Behaviour. **2011**,		22
195	Is iteration really easier to learn than recursion for CS1 students?. **2012**,		8
194	Automatic recognition of students' sorting algorithm implementations in a data structures and algorithms course. **2012**,		6
193	An eye-tracking study on the role of scan time in finding source code defects. **2012**,		62
192	Systematizing pragmatic software reuse. **2012**, 21, 1-44		44
191	Using Concept Maps to Assist Program Comprehension and Concept Location: An Empirical Study. **2012**, 11, 1250018		3
190	Categorizing variations of student-implemented sorting algorithms. **2012**, 22, 109-138		2

189	How do professional developers comprehend software?. **2012**,		81
188	Transactions on Edutainment VIII. *Lecture Notes in Computer Science*, **2012**,	0.9	1
187	Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. **2012**, 70, 143-155		40
186	Meta-Programming and Model-Driven Meta-Program Development. **2013**,		28
185	How Programmers Debug, Revisited: An Information Foraging Theory Perspective. *IEEE Transactions on Software Engineering*, **2013**, 39, 197-215	3.5	67
184	How Does Software Visualization Contribute to Software Comprehension? A Grounded Theory Approach. **2013**, 29, 743-763		2
183	An Information Foraging Theory Perspective on Tools for Debugging, Refactoring, and Reuse Tasks. **2013**, 22, 1-41		36
182	On the effect of program exploration on maintenance tasks. **2013**,		7
181	The impact of identifier style on effort and comprehension. *Empirical Software Engineering*, **2013**, 18, 219-276	3.3	59
180	Assessing the quality factors found in in-line documentation written in natural language: The JavadocMiner. **2013**, 87, 19-40		5
179	Meta-Program Development as a Model Transformation Process. **2013**, 189-208		1
178	Meta-Programming Task Specification Using Feature-Based Patterns and Domain Program Scenarios. **2013**, 171-188		1
177	Evaluating software clustering algorithms in the context of program comprehension. **2013**,		7
176	Program Comprehension. **2014**, 289-324		
175	. **2014**,		8
174	Human factors in webserver log file analysis. **2014**,		4
173	Degree-of-knowledge. **2014**, 23, 1-42		34
172	Developers' code context models for change tasks. **2014**,		22

171	Assessing representation techniques of programs supported by GreedEx. **2014**,	1
170	A visualization tool recording historical data of program comprehension tasks. **2014**,	2
169	Automatic Segmentation of Method Code into Meaningful Blocks: Design and Evaluation. **2014**, 26, 27-49	5
168	Visualizing the problem domain for spreadsheet users: A mental model perspective. **2014**,	
167	On the Comprehension of Program Comprehension. **2014**, 23, 1-37	55
166	Visualizing Software Structure Understandability. **2014**,	3
165	Enhanced JavaScript Learning Using Code Quality Tools and a Rule-Based System in the FLIP Exploratory Learning Environment. **2014**,	6
164	Human factors in software development: On its underlying theories and the value of learning from related disciplines. A guest editorial introduction to the special issue. **2014**, 56, 1537-1542	19
163	Eye Movements in Code Reading: Relaxing the Linear Order. **2015**,	52
162	Document Retrieval Metrics for Program Understanding. **2015**,	1
161	Task Mental Model and Software Developers Performance: An Experimental Investigation. **2015**, 36,	
160	Program comprehension with four-layered mental model. **2015**,	3
159	An empirical study on program comprehension task classification of novices. **2015**,	0
158	To fix or to learn? How production bias affects developers' information foraging during debugging. **2015**,	15
157	Code, Camera, Action: How Software Developers Document and Share Program Knowledge Using YouTube. **2015**,	31
156	Automatic comprehension of algorithms for algorithmic assessment. **2015**, 18, 413	1
155	The Impact of Hierarchies on the Architecture-Level Software Understandability - A Controlled Experiment. **2015**,	0
154	Comparing Trace Visualizations for Program Comprehension through Controlled Experiments. **2015**,	8

153	Tracing software developers' eyes and interactions for change tasks. **2015**,		61
152	Exploring Theory of Cognition for General Theory of Software Engineering. **2015**,		3
151	A mental model perspective for tool development and paradigm shift in spreadsheets. **2016**, 86, 149-163		4
150	Evaluation Experiences of the Representation Techniques of Greedy Programs: Application to the GreedEx Tool. **2016**, 11, 179-186		1
149	Foraging and navigations, fundamentally: developers' predictions of value and cost. **2016**,		7
148	Program Comprehension: Past, Present, and Future. **2016**,		19
147	Linguistic antipatterns: what they are and how developers perceive them. *Empirical Software Engineering*, **2016**, 21, 104-158	3.3	45
146	A theory of distances in software engineering. **2016**, 70, 204-219		30
145	Tracking Students' Cognitive Processes During Program Debugging: An Eye-Movement Approach. **2016**, 59, 175-186		25
144	Supporting comprehension of unfamiliar programs by modeling cues. **2017**, 25, 307-340		
143	Eye gaze and interaction contexts for change tasks: Observations and potential. **2017**, 128, 252-266		12
142	Documenting and sharing software knowledge using screencasts. *Empirical Software Engineering*, **2017**, 22, 1478-1507	3.3	10
141	Distributed analysis and filtering of application event streams. **2017**, 129, 1-25		
140	Shorter identifier names take longer to comprehend. **2017**,		22
139	Measuring neural efficiency of program comprehension. **2017**,		37
138	A problem posing-based practicing strategy for facilitating students' computer programming skills in the team-based learning mode. **2017**, 65, 1655-1671		24
137	Syntax, Predicates, Idioms - What Really Affects Code Complexity?. **2017**,		9
136	Comprehension First. **2017**,		60

135	On the use of visual clustering to identify landmarks in code navigation. **2017**,		0
134	Modeling information flow for an autonomous agent to support reverse engineering work. **2017**, 14, 245-256		1
133	Do Programmers do Change Impact Analysis in Debugging?. *Empirical Software Engineering*, **2017**, 22, 631-669	3.3	12
132	Software landscape and application visualization for system comprehension with ExplorViz. **2017**, 87, 259-277		21
131	Infusing Topic Modeling into Interactive Program Comprehension: An Empirical Study. **2017**,		1
130	Facilitating Scenario-Based Program Comprehension with Topic Models. **2017**,		
129	Beyond gaze. **2018**,		1
128	Simultaneous measurement of program comprehension with fMRI and eye tracking. **2018**,		19
127	Towards Understanding Programs by Counting Objects. **2018**,		
126	Search-Based Cost-Effective Software Remodularization. **2018**, 33, 1320-1336		2
125	Descriptive compound identifier names improve source code comprehension. **2018**,		12
124	Toward conjoint analysis of simultaneous eye-tracking and fMRI data for program-comprehension studies. **2018**,		5
123	Gaze as a Proxy for Cognition and Communication. **2018**,		3
122	A neuro-cognitive perspective of program comprehension. **2018**,		0
121	The GreedEx experience: Evolution of different versions for the learning of greedy algorithms. **2018**, 26, 1306-1317		3
120	Syntax, predicates, idioms □what really affects code complexity?. *Empirical Software Engineering*, **2019**, 24, 287-328	3.3	6
119	Shorter identifier names take longer to comprehend. *Empirical Software Engineering*, **2019**, 24, 417-443	3.3	8
118	Looks can mean achieving. **2019**,		0

117	Factors influencing dwell time during source code reading. **2019**,	2
116	On the Frequency of Words Used in Answers to Explain in Plain English Questions by Novice Programmers. **2019**,	1
115	Fifty years of the psychology of programming. **2019**, 131, 52-63	9
114	A Nano-Pattern Language for Java. **2019**, 54, 100905	2
113	Isopleth. **2019**, 26, 1-42	1
112	An Important and Timely Field. **2019**, 1-8	3
111	The History of Computing Education Research. **2019**, 11-39	12
110	Computing Education Research Today. **2019**, 40-55	2
109	Computing EducationLiterature Review and Voices from the Field. **2019**, 56-78	6
108	A Study Design Process. **2019**, 81-101	
107	Descriptive Statistics. **2019**, 102-132	2
106	Inferential Statistics. **2019**, 133-172	
105	Qualitative Methods for Computing Education. **2019**, 173-207	2
104	Learning Sciences for Computing Education. **2019**, 208-230	6
103	Higher Education Pedagogy. **2019**, 276-291	1
102	Engineering Education Research. **2019**, 292-322	3
101	Novice Programmers and Introductory Programming. **2019**, 327-376	19
100	Programming Paradigms and Beyond. **2019**, 377-413	11

99	Assessment and Plagiarism. **2019**, 414-444	2
98	Pedagogic Approaches. **2019**, 445-480	5
97	Equity and Diversity. **2019**, 481-510	5
96	Computational Thinking. **2019**, 513-546	14
95	Schools (K12). **2019**, 547-583	1
94	Computing for Other Disciplines. **2019**, 584-605	2
93	New Programming Paradigms. **2019**, 606-636	1
92	Tools and Environments. **2019**, 639-662	1
91	Tangible Computing. **2019**, 663-678	19
90	Leveraging the Integrated Development Environment for Learning Analytics. **2019**, 679-706	3
89	Teacher Learning and Professional Development. **2019**, 727-748	1
88	Learning Outside the Classroom. **2019**, 749-772	1
87	Student Knowledge and Misconceptions. **2019**, 773-800	1
86	Students As Teachers and Communicators. **2019**, 827-858	2
85	A Case Study of Peer Instruction. **2019**, 861-874	1
84	A Case Study of Qualitative Methods. **2019**, 875-894	
83	Index. **2019**, 895-906	
82	Cognitive Sciences for Computing Education. **2019**, 231-275	6

81	Teacher Knowledge for Inclusive Computing Learning. **2019**, 709-726		6
80	Motivation, Attitudes, and Dispositions. **2019**, 801-826		3
79	An Empirical Study Assessing Source Code Readability in Comprehension. **2019**,		1
78	Assessing students' understanding of object structures. **2019**,		0
77	Eye tracking analysis of computer program comprehension in programmers with dyslexia. *Empirical Software Engineering*, **2019**, 24, 1109-1154	3.3	4
76	Using reverse engineering techniques to infer a system use case model. **2019**, 31, e2121		
75	How to trick the Borg: threat models against manual and automated techniques for detecting network attacks. **2019**, 81, 25-40		1
74	Mining reading patterns from eye-tracking data: method and demonstration. **2020**, 19, 345-369		1
73	Characterizing the transfer of program comprehension in onboarding: an information-push perspective. *Empirical Software Engineering*, **2020**, 25, 940-995	3.3	3
72	Mining Association Rules from Code (MARC) to support legacy software management. **2020**, 28, 633-662		4
71	Exploring Programmers' API Learning Processes: Collecting Web Resources as External Memory. **2020**,		3
70	Using Hypotheses as a Debugging Aid. **2020**,		0
69	Do Programmers Prefer Predictable Expressions in Code?. **2020**, 44, e12921		1
68	A large scale empirical study of the impact of Spaghetti Code and Blob anti-patterns on program comprehension. **2020**, 122, 106278		5
67	How Developers Choose Names. *IEEE Transactions on Software Engineering*, **2020**, 1-1	3.5	2
66	There is No Such Thing as a □rial and Error Strategy□**2021**, 190-201		
65	Recording, Visualising and Understanding Developer Programming Behaviour. **2021**,		3
64	Striffs: Architectural Component Diagrams for Code Reviews. **2021**,		

63	Understanding large-scale software systems structure and flows. *Empirical Software Engineering*, **2021**, 26, 1	3.3	2
62	Considerations and Pitfalls in Controlled Experiments on Code Comprehension. **2021**,		0
61	Program Comprehension and Code Complexity Metrics: An fMRI Study. **2021**,		1
60	Is Algorithm Comprehension Different from Program Comprehension?. **2021**,		0
59	Exploring Reverse-tracing Questions as a Means of Assessing the Tracing Skill on Computer-based CS 1 Exams. **2021**,		0
58	EEG Activities During Program Comprehension: An Exploration of Cognition. *IEEE Access*, **2021**, 9, 120407-120421	3.5	1
57	Designing a Software Exploration Tool Using a Cognitive Framework. **2003**, 113-147		2
56	Reading Behavior and Comprehension of C++ Source Code - A Classroom Study. *Lecture Notes in Computer Science*, **2019**, 597-616	0.9	0
55	Investigating Eye Movements in Natural Language and C++ Source Code - A Replication Experiment. *Lecture Notes in Computer Science*, **2017**, 206-218	0.9	3
54	A Unified Ontology-Based Process Model for Software Maintenance and Comprehension. **2006**, 56-65		4
53	A Clustering-Based Approach for Tracing Object-Oriented Design to Requirement. **2007**, 412-422		3
52	Empowering Software Maintainers with Semantic Web Technologies. *Lecture Notes in Computer Science*, **2007**, 37-52	0.9	26
51	Automatic Quality Assessment of Source Code Comments: The JavadocMiner. *Lecture Notes in Computer Science*, **2010**, 68-79	0.9	26
50	Intelligent Software Development Environments: Integrating Natural Language Processing with the Eclipse Platform. *Lecture Notes in Computer Science*, **2011**, 408-419	0.9	4
49	Analysis of Experts and Novices Thinking Process in Program Debugging. *Communications in Computer and Information Science*, **2012**, 122-134	0.3	14
48	Effects of the Sequence of Game-Play and Game-Design on Novices Motivation, Flow, and Performance. *Lecture Notes in Computer Science*, **2012**, 46-55	0.9	2
47	Expert Systems and Creativity. **1987**, 173-193		6
46	The Tasks of Programming. **1990**, 45-62		23

45	Expert Programming Knowledge: A Schema-based Approach. **1990**, 205-222		8
44	Software Comprehension. **1988**, 107-121		3
43	Expertise and Instruction in Software Development. **1997**, 1105-1126		1
42	The Cambridge Handbook of Computing Education Research. **2019**,		18
41	Here we go again. **2020**,		8
40	A Human Study of Comprehension and Code Summarization. **2020**,		4
39	What Drives the Reading Order of Programmers?. **2020**,		3
38	Early childhood preservice teachers debugging block-based programs: An eye tracking study. *Journal of Childhood Education & Society*, **2020**, 1, 63-77	0.4	4
37	A Survey of Concepts Location Enhancement for Program Comprehension and Maintenance. *Journal of Software Engineering and Applications*, **2014**, 07, 413-421	0.6	3
36	Program Understanding Behavior During Estimation of Enhancement Effort on Small Java Programs. *Lecture Notes in Computer Science*, **2001**, 356-370	0.9	2
35	On the Meaning of Computer Programs. *Lecture Notes in Computer Science*, **2001**, 165-174	0.9	
34	Empirical Research on Program Comprehension.		
33	Design. **2005**, 139-168		
32	Personnel Management. **2005**, 51-71		
31	Programming. **2005**, 169-176		
30	Reverse Engineering Methods. **2007**, 47-65		
29	Measuring Cognition Levels in Collaborative Processes for Software Engineering Code Inspections. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, **2009**, 32-43	0.2	1
28	References. **2010**, 453-472		

27	An Experimental Study of the Logical Complexity of Data Structures. **1985**, 225-239		
26	Software Complexity Assessment and Human-Machine Interface Evaluation. **1987**, 187-202		
25	The Software Maintenance of Large Software Systems: Management, Methods and Tools. **1990**, 1-26		1
24	PROGRAM UNDERSTANDING AND KNOWLEDGE ORGANIZATION: THE INFLUENCE OF ACQUIRED SCHEMATA. **1990**, 245-256		2
23	A Schema-Based Model of Program Understanding. *Human Factors in Information Technology*, **1991**, 2, 225-239		0
22	Ein Visualisierungswerkzeug f□ die Wartung modularer Programme. *Informatik-Fachberichte*, **1991**, 405-414		
21	Program Comprehension Skills and Their Acquisition: A Call for an Ecological Paradigm. *NATO ASI Series Series F: Computer and System Sciences*, **1993**, 71-79		0
20	Ans□□ze des Programmverstehens. **1996**, 159-176		
19	Through (Tracking) Their Eyes: Abstraction and Complexity in Program Comprehension. *ACM Transactions on Computing Education*, **2022**, 22, 1-33	2.1	1
18	Performing Tasks Can Improve Program Comprehension Mental Model of Novice Developers. **2020**,		0
17	Interactive Explanation of Software Systems. **1997**, 53-75		1
16	BEACONS IN PROGRAM COMPREHENSION. *ACM SIGCHI Bulletin*, **1986**, 18, 56-57		3
15	On the cognitive development of the novice programmer. **2020**,		1
14	Exploring Algorithm Comprehension: Linking□Proof□and□Program□Code. **2021**,		
13	Naming Practices in Java Projects: An Empirical Study. **2021**,		0
12	What does this Python code do? An exploratory analysis of novice students□code explanations. **2021**,		
11	Pride: Prioritizing Documentation Effort Based on a PageRank-Like Algorithm and Simple Filtering Rules. *IEEE Transactions on Software Engineering*, **2022**, 1-1	3.5	1
10	An empirical evaluation of machine learning techniques to classify code comprehension based on EEG data. *Expert Systems With Applications*, **2022**, 117354	7.8	1

9	DeepCode: An Annotated Set of Instructional Code Examples to Foster Deep Code Comprehension and Learning. *Lecture Notes in Computer Science*, **2022**, 36-50	0.9	
8	Supporting program comprehension by generating abstract code summary tree. **2022**,		
7	Considerations and Pitfalls for Reducing Threats to the Validity of Controlled Experiments on Code Comprehension. *Empirical Software Engineering*, **2022**, 27,	3.3	1
6	How do we Help Students See the Forest from the Trees? **2022**,		
5	Pinpoint: A Record, Replay, and Extract System to Support Code Comprehension and Reuse. **2022**,		
4	Deja Vu: semantics-aware recording and replay of high-speed eye tracking and interaction data to support cognitive studies of software engineering tasks methodology and analyses. **2022**, 27,		0
3	Supporting program comprehension by generating abstract code summary tree. **2022**,		0
2	Unraveling novices code composition difficulties. 1-28		0
1	CrossCode: Multi-level Visualization of Program Execution. **2023**,		0