Exploratory and Live, Programming and Coding

A Literature Study

Patrick Rein[a], Stefan Ramson[a], Jens Lincke[a], and Robert Hirschfeld[a]

a Hasso Plattner Institute, University of Potsdam, Germany

Abstract Various programming tools, languages, and environments give programmers the *impression of changing a program while it is running*. This experience of *liveness* has been discussed since the 1950s and a broad spectrum of research on this topic exists. Amongst other, this work has been carried out in the communities around three major ideas which incorporate liveness as an important aspect: *live programming*, *exploratory programming*, and *live coding*.

While there have been publications on the focus of each single community, the overall spectrum of liveness across these three communities has not been investigated yet. Thus, we want to delineate the variety of research on liveness. At the same time, we want to investigate overlaps and differences in the values and contributions between the three communities.

Therefore, we conducted a literature study with a sample of 191 publications on the terms retrieved from three major indexing services. On this sample, we conducted a thematic analysis regarding the following aspects: motivation for liveness, application domains, modes of running a program, and types of contributions. We also gathered bibliographic information such as related keywords and prominent cited works.

Besides other characteristics the results show that the field of *exploratory programming* is mostly about technical designs and empirical studies on tools for general-purpose programming. In contrast, works on *live coding* have the most variety in their motivations and methodologies with a majority being empirical studies with users. As expected, most works on live coding are applied to performance art. Finally, work on *live programming* is mostly motivated by making programming more accessible and easier to understand, evaluating their tool designs through empirical studies with users.

In delineating the spectrum of work on liveness, we hope to make the individual communities more aware of the work of the others. Further, by giving an overview of the values and methods of the individual communities, we hope to provide researchers new to the field of *liveness* with an initial overview.

ACM CCS 2012

- *General and reference → Computing standards, RFCs and guidelines;*
- **Applied computing → Publishing;**

Keywords TODO

The Art, Science, and Engineering of Programming

Perspective The Empirical Science of Programming

Area of Submission TODO

1 Intro

A variety of programming environments and tools can provide the *impression of changing a program while it is running*[24, 27, 5]. Nowadays, this impression is often described as *liveness*. While research on *liveness* is not part of mainstream research on programming, still a broad spectrum of contributions exists. In particular, three ideas incorporate this capability as an integral part: *live coding*, *live programming*, and *exploratory programming*. As each term has its own research community, shared approaches and potential synergies might be overlooked. Further, the whole range of research on liveness only becomes visible when looking at all these communities. To delineate this potential across communities and illustrate the range of contributions, we conducted a literature study.

Looking at prominent publications from each community approximate differences become apparent. Generally speaking *live coding* is often concerned with the creation of art through changing source code as a performance in front of an audience [5, 2]. *Live programming* in contrast often seems to put the very activity of programming in its focus [23, 8]. Correspondingly, the term seems to be used when describing programming tools which provide immediate feedback on the dynamic behavior of a program even while programming. The term *exploratory programming* often refers to a particular workflow during programming whenever requirements are not fully defined but are yet to be discovered [27, 20]. It is supported by exploratory programming environments that incorporate changing a running system to make exploration of unknown domains or of design alternatives easier.

The communities around these ideas differ in their motives for dealing with changing a program while it is running, the fundamental perspective on the activity of programming, applications domains in which liveness is used, and desired outcomes of their research (system architectures, workflows, experiences). Besides their differences, they share the common notion of creating an impression of changing a program while it is running. Their differing approaches to this notion could be leveraged for cross-polination between the communities to advance all three of them. For example, *exploratory programming* systems could benefit from the specialized mechanisms and tools investigated under the term *live programming*. At the same time the *live programming* tools could be examined through the lense of experience reports from programmers as it is done in the *live coding* community.

All three communities are academic communities. Thus, we treat the publications published under the corresponding terms as artifacts which can be studied to gain insights into the values and practices of each community. Hence, we relate the communities around these terms to each other by surveying the similarities and differences in a sample of the corresponding literature. In particular, our contributions in this paper are:

- A systematically collected sample of 191 publications indexed in the ACM DL, IEEE Xplore, and DBLP on the terms *live programming*, *live coding*, and *exploratory programming*.[1]
- A thematic analysis [3] of the sample regarding the following aspects:
  - motivations for liveness
  - applications domains in which the liveness is used
  - types of contributions
  - historic distribution of terms
  - modes of running a program
  - prominent publications for each term
  - fields related to the three terms
- A comparison of the results of the thematic analysis between the corpus for each of the three terms

The remainder of the paper is structured as follows. We begin with the specific research questions guiding the literature study in section 2. We will then illustrate our methodology in section 3. We describe the characteristics of the three corpora in section 4 and the results of the study and the relation to the research questions in section 5. We discuss correlations in the study results, threats to validity, and potential consequences in section 6 and summarize our findings and point out further potential research in section 7.

2 Research Questions

The goal of this survey is to determine differences and commonalities between the three communities around the terms "live coding", "live programming", and "exploratory programming". In this context, we posed eight research questions.

1. The notion of an impression of changing a program while it is running has been around since the 1950s and the term *liveness* has been used since the 1990s [12, 24]. However, communities around these terms seem to have formed later, thus we want to determine *What is the distribution of publications on the different terms over time?*
2. While live coding seems to be mainly focused on programming for performance art, live programming and exploratory programming do not seem to have such a strong focus on one application domain. Therefore, we want to find out *In the context of which domains has liveness been used?*
3. The motivation for applying the technical concept for liveness can be telling about the values of the communities, thus we want to determine *What are the motivations for having liveness?*

[1] We will publish the bibliography files on Zenodo with the final version of this paper

4. In order to determine what kind of knowledge each community contributes we want to investigate *Which kinds of contributions are described?*
5. Besides the general motivation for liveness, the general reason for having a running program varies greatly between different approaches. For example, the Smalltalk community regards their systems as “live” as they are constantly running and any change to a Smalltalk program is actually a change in the object graph of the running system. In order to determine whether the three communities have a particular stance on this aspect we want to determine *What is the general reason for having a running version of the program while changing it?"
6. Each community has works which define their field. Assuming that these are the most cited works in their field we want to find out *What are the most prominent publications?*
7. The three terms used in this study are only one way of categorizing the field or research on programming. Simliar and overlapping perspectives might indicate further related work and future research potential. Thus, we want to determine *Which keywords have authors applied to their work?*

Finally, besides these research questions related to the publication we want to determine a list of artifacts providing liveness. These artifacts (such as tools, user interfaces, programming environments) created by the communities can be the foundation for determining common patterns in the design of a live programming experience.

3 Methodology

In order to determine the range of topics and compare the literature related to the three terms we conduct a thematic analysis on three corpora, one for each term. We did not aim to create a comprehensive systematic literature survey of the three terms but to determine the general distribution of themes of each corpus regarding to our research questions.

In general, we followed the SALSA (Search, AppraisaL, Synthesis, Analysis) process [7]. First, we created an overall corpus with a systematic search and an appraisal of works. We then synthesized data for the individual research questions by extracting data from the publications and by doing a thematic analysis [3]. We analyzed the resulting data by determining correlations, comparing overlaps, and differences between the corpora defined by the three terms.

3.1 Search: Systematically Determined Corpora

For each term we created an initial corpus and then created the hull of references. We determined the initial corpora by using the full-text search on three major indexing services :

- IEEE Xplore (http://ieeexplore.ieee.org/search/searchresult.jsp?queryText=.QT.KEYWORD.QT.&newsearch=true)

- ACM Digital Library (ACM DL): Guide to Computing (http://dl.acm.org/exportformats_search.cfm?query=KEYWORD&filtered=&within=owners%2Eowner%3DGUIDE&dte=&bfr=&srt=%5Fscore&expformat=bibtex)
- Digital Bibliography & Library Project (DBLP) (http://dblp.uni-trier.de/search/publ/api?q=KEYWORD%24&h=1000&format=bib1&rd=1a)

We retrieved these search results through an automated process using the URLs denoted in brackets (insertion of the keyword denoted as KEYWORD). Depending on the capabilities of the search API the keywords were either entered as exact matches (for example "live programming" at IEEE) or AND queries (for example live+programming for DBLP).

Further, we manually added publications from manually selected scientific venues (For a detailed list of keywords and venues per term, see table 1). Namely, we added the International Conference on Live Coding (ICLC) and the Live Programming Workshop (LIVE). Other major venues, such as CHI, UIST, or PLATEAU are fully indexed by the three services we used and thus appeared automatically through our initial search. We then selected appropriate publications from this initial corpus using our appraisal criteria.

Term	Keywords	Venues
live coding	livecoding, live coding	ICLC 2015, ICLC 2016, ICLC 2017[2]
live programming	live programming	LIVE 2013, LIVE 2016, LIVE 2017
exploratory programming	exploratory programming	-

3.2 Appraisal

The appraisal criteria are concerned with the nature of the publication as well as the content of publications. The complete appraisal process was done manually.

For a publication to be accepted it has to be peer-reviewed. This covers publications for conferences, workshops, and journals, as well as doctoral theses. To ensure that the papers contained enough information for us to evaluate the content, we removed all publications with only one page.

In general, a publication is added to a term-specific corpus if it contains the corresponding term in the title, keywords, abstract or the title of the publication it is part of, for example the title of the proceedings. Further, whenever the term occurred we manually determined whether it actually referred to the particular experience of programming we want to investigate. This is necessary as the three terms live programming, live coding, and exploratory programming are also used in other domains. Consequently, we only selected papers adhering to the following notion of programming:

[2] Only contains the publications which were actually made available from the conference website.

> "the human activity of describing a process run by a computer"

This excludes the following domains from the corpora:

- planning and modification of television schedules
- encoding video streams during recording and broadcasting
- algorithms for artifical intelligence which explore a solution space
- planning health programs
- analysis of concurrent programs for the liveness property

Further, the term "live coding" is used in *teaching research* to refer to a class room constellation in which teachers program "live" in the sense that students can follow the teachers' activities in a programming environment on a projector. We also excluded publications on this topic as they do not include an explict notion of changing a program while it is running. The fact that the mere writing of source code is done "live" in front of an audience (without continuously executing the program) those not entail any particular connection between the code and its dynamic behavior.

Also, while exploratory programming environments refers to environments which often include some form of liveness, the single term "exploratory programming" can also refer to a particular workflow [27]. This "exploratory workflow" is creating new alternatives and trying them out. This does not entail liveness but can also be done in an edit-compile-run cycle with a long roundtrip time. We excluded papers exclusively referring to exploratory workflows and did not mention exploratory programming environments. [This list comprised 10 papers]

3.3 Synthesis: Thematic Analysis

We derived the data for research questions 1, 6, and 7 directly and manually from the individual publications. The specific procedures used are reported in [results].

As the goal of the study is to illustrate the spectrum of research on the topic of liveness, we applied thematic analysis instead of a pre-defined code book to gather data for research questions 2 to 4 [3]. That means we determined the codes for the themes application domain, motivation, and types of contribution during coding. Specifically, we used theoretical thematic analysis with the themes defined by our research questions [3]. The codes for the theme modes of running were determined beforehand.

All publications were coded as one corpus in order to prevent bias of the coder towards codes who they might deem fitting to the corpus a publication belongs to. The codes are described in detail in the related sections in [results]. All coding was done by one coder. During the first pass the set of codes changed, as codes were added or the focus of a code changed. Thus, we did a second pass through the corpus for each theme revising the codes.

Afterwards, the coder formalized the codes in the descriptions. We determined the inter-coder reliability for a 20% sample using Cohen's kappa. We report the inter-coder reliablity with each result.

3.4 Analysis

The main goal of the study is to illustrate the spectrum of research on liveness. Thus, we directly report the determined codings for the overall corpus. We do not report on the distribution of codes for the overall corpus, as the three terms are disproportionally represented and consequently the overall distribution would be skewed to the distribution of the term with the most publications.

Another goal of the study is to investigate the overlap and differences of the terms *live programming*, *live coding*, and *exploratory programming*. Hence, as a further analysis we compare the distribution of the codes between the three term-specific corpora.

4 Corpus Characteristics

4.1 Selection

From how many papers to how many papers?

4.2 Publication Data

4.2.1 Overlap

4.2.2 Venues and Journals

Term	Venues / Journals (sorted alphabetically)
live programming	CHI, CHI, COP, ComposableWeb, ECOOP, FARM@ICFP, HCC, ICSE, ISMM, LIVE, MOVILESoft, OOPSLA, Onward!, PLDI, PROMOTO, PX, Programming Journal, SIGMOD, SPLASH, UIST, VL/HCC, VRST
exploratory programming	CHI, COMPCON, ECOOP, HOPL, ICSENG, IMCSIT, ISCA, OOPSLA, Onward!, PX, VL, VL/HCC
live coding	AM, Blocks and Beyond, C&C, CLEI, DUXU, EVA, EvoMUSART, FARM, ICFP, ICLC, ICSE, LIVE, MM, NIME, OZCHI, SIGCSE, TEI, VL/HCC

4.2.3 Distribution over Time

4.2.4 Keywords

In order to determine other perspectives on the field of research on programming related to liveness, we extracted the keywords of each publication. We only recorded keywords mentioned in the actual publication. We also only recorded the keywords chosen by the authors and ignored general tags and categories pre-determined by the publisher (for example the ACM subject classification).

We cleaned the data marginally by merging the keywords “programming environment” and “programming environments” as well as “end-user development” and “end-user programming”. Finally, we selected all keywords mentioned more then once.

Term (works with terms / corpora size)	Keywords (count)
live programming (50/72)	live programming (32), live coding (6), end-user programming (5), debugging (4), direct manipulation (4), Smalltalk (4), live programming environment (3), music (3), mashups (3), spreadsheets (3), programming environment (3), Datalog (3), web services (2), prototyping (2), integrated development environment (2), data analysis (2), incremental maintenance (2), streaming data (2), LogiQL (2), testing (2), LogicBlox (2), Javascript (2), liveness (2), mashup tools (2), data mining (2)
exploratory programming (12/27)	exploratory programming (6), programming environment (4), Self (2), direct manipulation (2), gestures (2), liveness (2), programming by demonstration (2), virtual machine (2), live programming (2), Forms/3 (2)
live coding (29/101)	live coding (22), music (6), generative art (4), software engineering (2), visual programming (2), liveness (2), art (2), domain specific languages (2), creative coding (2), generative music (2), programming environment (2), functional programming (2), music performance (2), web audio (2), improvisation (2), visualization (2)

5 Study Results

For each research question state: - coding scheme - inter coder reliability - results

Three results regarding the liveness described in the publications, One result regarding research approach

5.1 Which domains was the liveness used in (specifically, if it was only given as an example it should not be counted)? (not mutually exclusive)

5.1.1 Coding Scheme

To answer the question for which application domains liveness plays a role in each corpus, we coded the intended application domain described in a publication. Specifically, we coded the domain of application for which the *liveness* was relevant. Further,

the domain had to be a major theme. We did not code domains which were only mentioned as examples. The codes are not mutually exclusive.

Education and training The program is created to either learn programming or to learn something from the program (for example a model of ecological processes).

Simulation Systems and programs for simulation

Data analysis The analysis of existing data to gain insights

Visual performance art The in-time creation of visual art in front of an audience or for recording.

Audio performance art The in-time creation of audible art in front of an audience or for recording.

Performance art Any other form of performance than pure visuals and audio, for example writing poems, dance, general performance art reflections

Web development The construction of applications running in or on the web. This does not involve the construction of programs through an environment implemented as a web application.

User interfaces The design and implementation of user interfaces.

Databases The programming and manipulation of databases.

Programming languages The design and implementation of programming languages (syntax and semantics).

Systems programming Development of execution environments or fundamental libraries

Computer graphics Creating programs concerned with generating visual output (though not as a performance in contrast to visual performance art)

General software development General software development with no focus on a particular application domain.

Artifical intelligence The design and implementation of artifical intelligence systems.

Physical computing All systems which have some interface with the physical world (actuators or sensors).

Virtual/Augmented reality

Enterprise-Resource-Planning (ERP) systems

Computer games

Software verification

Text editing

5.2 Intended Outcome

TODO: outcome is not the right word TODO: explain program vs system in more detail: system is about modifying the runtime state which in turn might represent a program TODO: Jens put it like: What is the goal of the user of the meta system? The notion of “an impression of changing a program while it is running” subsumes different modes of “running” a program.

5.2.1 Coding Scheme

We distinguish three modes of "running a program" according to the intended outcome:

Computing a result The programmer wants some output and therefore runs a program. The program is only secondary and might be discarded after the computation, this covers in-time output such as generated sound as well as discrete outputs created by a data analysis algorithm.

Creating a program The programmer wants to create a program. The program is currently running to get feedback on the dynamic behavior of it but the resulting artifact of the activity are some static representation of the program behavior.

Evolving a system The programmer wants to evolve a system which is constantly running. Thus, programmers change behavior descriptions as well as the state of the system. Examples are image-based systems or database systems.

Notably, all three modes of running a program are possible for most activities and programs (for example a programmer might keep a Python REPL for data analysis open for a long time). Therefore, we assigned the code only to the emphasis in the described work. Although, the categories are mutually exclusive, we allowed assigning multiple codes to a publication when it described multiple aspects of the activity of the programmer.

5.3 Motivation for the term stated

While liveness is generally an intriguing idea, the motivations for it vary greatly between authors and communities. However, the initial motivation influences designs and research methodology and is thus an integral element of the definition of a field.

5.3.1 Coding Scheme

The following coding scheme reflects the variety of actual motivations for liveness. When determining the motivation in a publication we only assigned a code when it was a) explicitly stated or b) a fundamental theme of the work. As we extracted the list directly from the publications, the items do not always have the same level of abstraction.

Productivity Liveness improves productivity, most often through making some activity during programming faster

Creativity This applies to all works specifically mentioning creativity as their primary motication.

Exploration of alternatives This is the very specific notation of livness enabling programmers to explore distinct alternative solutions (It does not suffice to say it supports "exploratory programming" as this would often be self-referential; mentioning "exploratory work" falls in this category though)

Live tuning Liveness as a means to explore a continuous parameter space of a design

Comprehension Ease the process of comprehension and learning of program behavior or the program domain

In-time output Livness is an integral part of the activity itself. The output has to be created and modified with a short delay as the changing of the output is the relevant action; this does often correlate with live coding but does not have to. <!– There are argumentations which argue that industrial live programming is similar due to pair programming –>

Collaboration The liveness enables collaboration between users because others can see the effects of owns changes

Accessibility Liveness makes programming more accessible. This involves comprehension but is more specific towards enabling less experience programmers

Engagement Getting users more enganged through liveness

Enstrangement Live programming as an extreme form of programming which enables reflections on the nature of programming

Can not be determined We added this code explictly in order to determine how often liveness is taken as granted.

Again, these codes are not mutually exclusive.

5.4 What type of contribution is described?

As illustrated before, the three communities differ greatly in their motivatios for liveness and desired outcomes. These different world views also reflect in the inquiries and the contributions communicated within the community.

5.4.1 Coding Scheme

We code the contribution of each publication. Some of these codes are hierarchical so a code can have several sub-codes which also count in the super-code set (see for example the empirical codes).

Empirical studies Works which investigated an existing artifact or phenomena. As this covers a wide range of contributions we made codes more specific with the following additional information: a) fixed / flexible setup[3] b) quantitative / qualitative outcome c) humans as subject / systems as subjects

Survey Surveys of existing artifacts, for example publications or systems

Technical design The description of a technical implementation

User interface design Describes the design of a particular user interaction. This can be a novel form of interaction, a new input device, or a new way of output provided by the system.

Programming interface design This involves programming language design (syntax and semantics) as well as the design of the interface of libraries.

Tool design The description of the design of a programming tool or environment

Individual account of experience An individual subjective account

[3] Was the methodology completely determined before the observation or was it adapted during the observation [TODO: cite real world research]

Formal methods The description of a formal model or the application of an existing model

Algorithm design The description of an algorithm

Practices The description of practices either empirically observed or from subjective experience

Design principles A collection of rules or values for designs

Essay An application of theoretical models or principles to an existing phenomena [TODO: wat?]

Performance The description of an artistic performance

Curriculum design The description of a new curriculum or teaching model

5.5 Prominent Publications

In order to determine the which publications define and form each field, we determined the most prominent publications referenced by a corpus. Therefore, we collected all references from publications of one corpus. The cited publication did not have to be in the corpus itself in order to be collected. We list the five most cited references. If there are multiple publications with the same citation count at position five we included all publications with that citation count.

5.5.1 Exploratory Programming

- (9) Smalltalk-80: The Language and Its Implementation [6]
- (7) Smalltalk and exploratory programming [19]
- (5) Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems [10]
- (4) Self: The Power of Simplicity [21]
- (4) The Design and Implementation of the Self Compiler, an Optimizing Compiler for Object-Oriented Programming Languages [25]
- (4) An Efficient Implementation of SELF - a Dynamically-Typed Object-Oriented Language Based on Prototypes [1]
- (4) Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself [9]

5.5.2 Live Programming

- (18) Real-time Programming and the Big Ideas of Computational Literacy [17]
- (14) Smalltalk-80: The Language and Its Implementation [6]
- (13) VIVA: A Visual Language for Image Processing [24]
- (12) Living It Up with a Live Programming Language [14]
- (11) Usable Live Programming [15]

5.5.3 Live Coding

- (26) Live Coding in Laptop Performance [5]

- (12) The programming language as a musical instrument [2]
- (12) Gibber: Live Coding Audio in the Browser [18]
- (11) Live Coding of Consequence [4]
- (11) Aa-Cell in Practice: An Approach to Musical Live Coding [22]
- (11) Rethinking the Computer Music Language: SuperCollider [13]
- (11) Visualisation of Live Code [16]
- (11) The IXI Lang: A SuperCollider Parasite for Live Coding [11]
- (11) Tidal - Pattern Language for the Live Coding of Music [26]

6 Discussion

6.1 Is there an Exploratory Programming Environments Community?

- Exploratory programming: yes!
- exp prog envs: Not clear
- Smalltalk and Self in keywords suggests its particular language communities

6.2 Threats to Validity

- Missing hull: For example, the original Viva paper is missing
- Selection based on indexing services
- Threats to validity: Indexing services + OCR and thereby maybe wrong positive

7 Conclusion

A Detailed Data Sets

A.1 lpsSynthesisDomains

codes	all	live prog.	expl. prog.	live coding
Artifical intelligence	3	0	2	1
Audio performance art	78	4	0	78
Computer graphics	11	7	0	6
Data analysis	15	11	3	3
Database development	5	5	0	0
Education and training	24	10	2	16
Enterprise-Resource-Planning (ERP) systems	3	3	0	0
Game development	1	1	0	0
General software development	70	45	17	13
Performance art	15	0	0	15

Exploratory and Live, Programming and Coding

codes	all	live prog.	expl. prog.	live coding
Physical computing	7	5	0	2
Programming languages	5	4	0	1
Simulation	6	2	4	0
Software verification	2	2	0	0
Systems programming	7	2	3	2
Text editing	1	0	0	1
User interfaces	13	7	4	2
Virtual/Augmented reality	4	2	0	2
Visual performance art	21	3	0	20
Web development	6	6	0	0
Total	297	119	35	162

A.2 lpsSynthesisIntendedEffect

codes	all	live prog.	expl. prog.	live coding
Computing a result	101	18	3	86
Creating a program	51	35	6	13
Evolving a system	43	23	16	5
Not applicable	6	0	2	4
Total	201	76	27	108

A.3 lpsSynthesisMethodOfInquiry

codes	all	live prog.	expl. prog.	live coding
Algorithm design	6	2	0	4
Can not be determined	3	0	2	1
Curriculum design	4	1	1	3
Design principles	11	5	3	4
Empirical study (system)	27	13	8	6
Empirical study (users)	36	16	10	17
Essay	20	2	0	18
Formal methods	9	8	0	1
Individual account of experience	14	0	2	12
Performance	19	2	0	19
Practices	11	0	0	11
Programming interface design	37	15	7	18
Survey	3	1	0	3
Technical design	75	28	14	36
Tool design	83	50	11	26
User interface design	34	15	3	18

codes	all	live prog.	expl. prog.	live coding
Total	392	158	61	197

A.4 lpsSynthesisMotivation

codes	all	live prog.	expl. prog.	live coding
Accessibility	23	19	2	3
Can not be determined	33	14	8	11
Collaboration	17	1	1	15
Comprehension	25	22	1	5
Creativity	3	0	1	2
Engagement	4	1	0	3
Enstrangement	2	0	0	2
Exploration	32	15	11	7
In-time output	81	8	0	78
Live tuning	5	2	0	3
Productivity	17	8	6	6
Total	242	90	30	135

References

[1] *An Efficient Implementation of SELF - a Dynamically-Typed Object-Oriented Language Based on Prototypes*.

[2] Alan Blackwell and Nick Collins. "The Programming Language as a Musical Instrument". In: *Proceedings of PPIG05 (Psychology of Programming Interest Group)* 3 (), (284 to: 289).

[3] Virginia Braun and Victoria Clarke. "Using Thematic Analysis in Psychology". In: *Qualitative research in psychology* 3.2 (), (77 to: 101).

[4] Nick Collins. *Live Coding of Consequence*.

[5] Nick Collins, Alex McLean, Julian Rohrhuber, and Adrian Ward. *Live Coding in Laptop Performance*.

[6] Adele Goldberg and David Robson. *Smalltalk-80: The Language and Its Implementation*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-11371-6.

[7] Maria Grant and Andrew Booth. "A Typology of Reviews: An Analysis of 14 Review Types and Associated Methodologies". In: *Health Information & Libraries Journal* 26.2 (), (91 to: 108).

[8] Christopher Hancock. "Real-Time Programming and the Big Ideas of Computational Literacy". PhD thesis.

[9] Daniel Ingalls, Ted Kaehler, John Maloney, Scott Wallace, and Alan Kay. "Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself". In: *Smalltalk and Exploratory Programming*. Volume 32. 10. Atlanta, Georgia, USA: ACM, (318 to: 326). DOI: 10.1145/263698.263754.

[10] Henry Lieberman. "Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems". In: *Proceedings of the Conference on Object-oriented Programming Systems, Languages and Applications (OOPSLA) 1986*. OOPSLA '86. Portland, Oregon, USA: ACM, (214 to: 223). ISBN: 0-89791-204-7. DOI: 10.1145/28697.28718.

[11] Thor Magnusson. "The IXI Lang: A SuperCollider Parasite for Live Coding". In: *Proceedings of the Computer Music Conference (ICMC) 2011*. Michigan Publishing.

[12] John McCarthy. "History of LISP". In: *SIGPLAN Not.* 13.8 (), (217 to: 223). ISSN: 0362-1340. DOI: 10.1145/960118.808387.

[13] James McCartney. "Rethinking the Computer Music Language: SuperCollider". In: *Computer Music Journal* 26.4 (), (61 to: 68).

[14] Sean McDirmid. "Living It Up with a Live Programming Language". In: *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) 2007*. Volume 42. OOPSLA '07 10. Montreal, Quebec, Canada: ACM, (623 to: 638). ISBN: 978-1-59593-786-5. DOI: 10.1145/1297027.1297073.

[15] Sean McDirmid. “Usable Live Programming”. In: *Proceedings of the Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!) 2013*. Onward! 2013. Indianapolis, Indiana, USA: ACM, (53 to: 62). ISBN: 978-1-4503-2472-4. DOI: 10.1145/2509578.2509585.

[16] Alex McLean, Dave Griffiths, Nick Collins, and Geraint Wiggins. “Visualisation of Live Code”. In: *Proceedings of Electronic Visualisation and the Arts*.

[17] *Real-time Programming and the Big Ideas of Computational Literacy*.

[18] Charles Roberts and JoAnn Kuchera-Morin. “Gibber: Live Coding Audio in the Browser”. In: *Proceedings of the Computer Music Conference (ICMC) 2012*. Michigan Publishing.

[19] D Sandberg. *Smalltalk and exploratory programming*.

[20] BA Sheil. *Power Tools for Programmers*.

[21] Randall Smith and David Ungar. *Self: The Power of Simplicity*. Technical report 12. Orlando, Florida, USA, (227 to: 242). DOI: 10.1145/38765.38828.

[22] Andrew Sorensen and Andrew Brown. “Aa-Cell in Practice: An Approach to Musical Live Coding”. In: *Proceedings of the Computer Music Conference (ICMC) 2007*. Michigan Publishing.

[23] Steven Tanimoto. “A Perspective on the Evolution of Live Programming”. In: *1st International Workshop on Live Programming (LIVE 2013)*. LIVE ’13. San Francisco, California: IEEE Press, (31 to: 34). ISBN: 978-1-4673-6265-8. DOI: 10.1109/LIVE.2013.6617346.

[24] Steven Tanimoto. “VIVA: A Visual Language for Image Processing”. In: *Journal of Visual Language Computation* 1.2 (), (127 to: 139). DOI: 10.1016/S1045-926X(05)80012-6.

[25] *The Design and Implementation of the Self Compiler, an Optimizing Compiler for Object-Oriented Programming Languages*.

[26] *Tidal - Pattern Language for the Live Coding of Music*.

[27] Jason Trenouth. “A Survey of Exploratory Software Development”. In: *The Computer Journal* 34.2 (), (153 to: 163).

About the authors

Patrick Rein Contact him at patrick.rein@hpi.uni-potsdam.de.

Stefan Ramson Contact him at stefan.ramson@hpi.uni-potsdam.de.

Jens Lincke Contact him at jens.lincke@hpi.uni-potsdam.de.

Robert Hirschfeld Contact him at robert.hirschfeld@hpi.uni-potsdam.de.