# Building a Corpus of Task-Based Grading and Feedback Systems for Learning and Teaching Programming

https://doi.org/10.3991/ijep.v12i5.31283

Sven Strickroth[1](✉), Michael Striewe[2]
[1]Ludwig-Maximilians-Universität München, Munich, Germany
[2]University of Duisburg-Essen, Essen, Germany
sven.strickroth@ifi.lmu.de

**Abstract—**Using grading and feedback systems in the context of learning and teaching programming is quite common. During the last 20 to 40 years research results on several hundred systems and approaches have been published. Existing papers may tell researchers what works well in terms of educational support and how to make a grading and feedback system stable, extensible, secure, or sustainable. However, finding a solid basis for such kind of research is hard due to the vast amount of publications from a very diverse community. Hardly any recent systematic review includes data from more than 100 systems (most include less than 30). Hence, the authors started an endeavor to build a corpus of all task-based grading and feedback systems for learning and teaching programming that deal with source code and have been published in recent years. The intention is to provide the community with a solid basis for their research. The corpus is also designed to be updated and extended by the community with future systems. This paper describes the process of building the corpus and presents some meta-analysis that shed light on the involved research communities.

**Keywords—**e-assessment, programming, automated assessment, feedback system, submission system, tutoring system, learning environment, automated grading, corpus of systems, computer science education

## 1 Introduction

Programming education is a curricular element in many disciplines in higher education and schools [1] but it is not easy to acquire programming skills [2]: Learners must solve many programming tasks in order to transform theoretical knowledge into practical skills. They often face immense obstacles when trying to create, test and debug their (first) programs. Hence, support and feedback play a central role in the learning process. In formal contexts teachers can directly support their students. However, teachers may lack enough time to address the needs of all students in a timely and detailed manner. Therefore, it is common to automate (at least) parts of the analysis of learners' solutions as well as the provision of feedback. The analysis results can be passed automatically to the learner in order to provide consistent and accurate

feedback (cf. [3]) or also to the teacher for an overview over the class or semiautomatic assessment that can offer a holistic assessment since points can be manually awarded for minor errors (cf. [4]).

Research and development on support systems for programming education has a considerable long history. First approaches for automated grading date back at least to the 1960s [5]. Since then, researchers published results on several hundred systems and approaches (cf. Section 2). The number of unpublished systems is certainly even larger, since computer science departments or individual teachers may have implemented their own systems without having the opportunity or interest to share their solution in research or practice communities via formal publications. But even among the published systems, many of them have very similar features and could basically be used for an assessment in other scenarios [6]. However, the large number of existing, similar systems seems to tell a different story. Two main reasons for the diversity of systems could be identified [6, 7]: (1) A system was developed for a specific course. Significant features of the system are tailored exactly for that course, limiting its easy reuse in other contexts. Adaptation to other contexts would often be possible but is not realized due to a lack of e. g., resources, knowledge about existing systems, or cooperation. (2) A system was developed in the context of a thesis or research project with a focus on the creation and evaluation of a new approach. Systems created in that way are usually prototypes that do not receive support for sustainable and versatile use in practice after the project or thesis has been completed.

Moreover, some research or practice communities use their own terminology to characterize systems and name the problems they tackle (cf. Section 4.4). Such systems may stay hidden for researchers who use different terms and names. These reasons (or the plain lack of opportunity or interest in publishing systems and implementation details) may cause repeated development of similar systems and hinder systems from becoming widely accepted (cf. [8]).

Therefore, we started an endeavor to build an extensible corpus of existing task-based grading and feedback systems that can be used by other researchers and practitioners. The corpus aims to achieve a maximum coverage of the current state of the art on grading and feedback systems. It employs a multi-step systematic search approach that includes an analysis of existing reviews, trawling through major publishing venues, and snowball search on references (cf. Section 3). The corpus can be found on https://systemscorpus.strickroth.net. By design, the corpus will virtually never be complete but is meant to be verified, updated, and extended by the research community as new papers are published. The construction process so far covered several international publication venues as well as one national community and focuses on systems published 2008 or later. It is intended to cover more venues, national communities and older publications in the future.

The contributions of the article are an overview of existing reviews, a description of the procedure to construct a reusable and extendable corpus of task-based grading and feedback systems for programming education, a characterization of the current version of the corpus, and hints on how to use the corpus as a research tool.

The remainder of this paper is structured as follows: Related work, i. e. other reviews and surveys, is presented in Section 2. The search method and selection policy are described in Section 3. A short meta-analysis of the corpus is performed to highlight its advantage in contrast to existing surveys in Section 4. A discussion of the achievements as well as directions for future research and work on the corpus closes the paper in Section 5.

## 2 Related work

The large number of existing review and survey papers [2, 4, 6, 7, 9–30] (non-exhaustive list), shows the long history of research and development on support systems for programming education in recent decades. The existing reviews can be categorized as follows:

- Reviews focusing on introductory programming literature in general [2, 7]. These overviews mention specific grading and feedback systems only as examples.
- Reviews focusing on specific topics such as visualization approaches (e. g., [20]), programming environments (e. g., [14, 15]), intelligent tutoring systems (e. g., [13]), hint systems (e. g., [30]), or static analysis techniques (e. g., [23]). These reviews consider actual systems primarily as vehicles to apply the respective approaches and often omit technical details.
- Reviews providing an overview on existing systems for supporting programming education in some way – possibly including some classification of systems based on the approaches they use or other characteristics like the target group or the programming paradigms they support.

Most relevant for this paper is the latter category. Table 1 provides an overview and summary of the existing reviews of that type that have been analyzed for this paper. The number of analyzed systems ranges from 11 to 101 in these reviews. Unfortunately, for a majority of reviews the selection criteria for papers and systems are unclear. Hence, it is hard to judge whether each review provides a good coverage of the relevant systems for the respective context. Notably, no review aims for completeness [24, 27] but a sufficient coverage is a prerequisite for robust results. At least there seems to be a trend to more systematic surveys.

All existing reviews only include systems published in English. However, there is evidence that there are strong local communities with non-English publications (cf. Section 4.2). Notably, the only meta-review that compares existing reviews based on the terminology and categorization they use and that is known to the authors is also not published in English [31].

**Table 1.** Focus of existing review and survey papers with a focus on tools, based on an analysis in [31]

| Paper | Year | Focus/Classification | Methodology | # Systems |
|---|---|---|---|---|
| [12] | 2001 | classification of web-based systems based on pedagogical approaches and critical analysis | unclear | 22 |
| [4] | 2005 | automatic assessment methods; also discusses semi-automatic vs. automatic as well as formative vs. summative assessment | unclear | 23 |
| [17] | 2005 | historical overview; also identifies general difficulties and problems of automatic assessment | "text search of ACM journals and conferences" | 14 |
| [16] | 2005 | review and classification of cutting-edge and innovative approaches and tools | unclear | 20 |
| [18] | 2009 | dynamic and static analysis approaches; also names advantages, disadvantages, and directions of future development | unclear | 22 |
| [6] | 2010 | system features; also covers aspects such as security, licensing and availability | searching phrases on ACM Digital Library and IEEE Xplore | 17 |
| [19] | 2012 | interoperability of programming evaluation tools | unclear | 15 |
| [21] | 2013 | evaluation metrics; rough classification into "mature" and "recently developed" tools | unclear | 11 |
| [22] | 2013 | AI-supported tutoring approaches; also identifies types and techniques | unclear | 17 |
| [25] | 2016 | adaptive feedback approaches | specifies a systematic approach | 20 |
| [24] | 2016 | classification on (semi-)automatic assessment type, student or teacher centered approaches and speciality (contest, quiz, software testing) | Systematic Literature Review (cf. [32]) | 30 |
| [27] based on [26] | 2018 | formative feedback approaches; also analyses adaptability and quality of evaluations | Systematic Literature Review (cf. [32]) | 101 |
| [28] | 2018 | static/dynamic analysis approaches, strengths and limitations | unclear | 17 |
| [29] | 2019 | assessment approaches and grading tools | unclear | 17 |

## 3 Paper and system selection

In order to produce reliable and conclusive contents for the corpus, both a set of criteria for inclusion or exclusion of systems as well as a procedure for retrieving and judging papers must be defined.

### 3.1 Inclusion criteria

As a consequence of building a corpus of systems, we defined inclusion criteria for systems and not for individual papers. Hence, the primary decision is on whether a system is included or excluded. Gathering and analyzing all available papers on a particular system is a different task. To be included in the corpus, a system must meet all of the following requirements:

1. The system must present tasks (e. g., homework exercises or exam assignments), accept solution submissions and provide automated feedback that is more sophisticated than typical compiler or interpreter messages. Systems that assist in explorative learning by giving feedback, or creating visualizations but without presenting actual tasks are not considered for the corpus. Pure peer assessment systems, web IDEs, or middlewares also are not considered.
2. The system must request students to enter programming source code and must allow students some degree of design choices, e. g., in how they realize an algorithm. Systems that only use simple fill-in-the-gap items in which each gap does not require more than a single expression, statement, or line are not considered for the corpus.
3. The system must deal with source code for programming languages. Systems for declarative query languages (e. g., SQL), markup languages (e. g., HTML), or block-based programming approaches (e. g., Scratch) are not considered. However, the latter may be included in future versions of the corpus.
4. The system must have been fully implemented at least for prototypical or experimental use. Papers "just" describing an approach or algorithm without indicating itself as an implementation ("system", "tool", etc.) are excluded.
5. At least one scientific paper on the system must have been published in 2008 or later. Systems that are mentioned only in surveys or as related work within this period are not considered for the corpus. There are no formal requirements for papers (such as being longer than two pages), as long as they contain sufficient information to decide about inclusion.

There are no requirements on whether a system is intended to be used in formative or summative assessments, whether it works online or offline, and whether it is intended for classroom use (i. e. including reporting results to a teacher) or pure self-assessment.

To illustrate the criteria, we discuss some corner cases of inclusion and exclusion: Plain web-based development environments such as 5code [33] or editors providing more elaborated error messages such as Decaf [34] are excluded due to the first requirement as they are not task-based. JAssess [35], a tool in which compilation is performed automatically, also misses the first requirement as no further automatic feedback beyond compiler messages is provided to students. The second requirement is not fulfilled by tools such as C-doku [36] with only mini fill-in-the-gap tasks. However, this requirement also brought up border cases such as the systems INCOM [37] and J-LATTE [38] – both have pre-structured input fields, however, provide all freedoms regarding the algorithm to the student and are, therefore, included. A prominent example of a system which is not included in the corpus due to the fifth requirement is MARMOSET [39]: This system was developed around the year 2005, is cited in many (review) articles, and was still in development until 2015 (when GoogleCode

was shut down). However, no scientific publications for that system could be found that were published after 2006. Also, most commercial systems are excluded, because those were not presented to the scientific community.

### 3.2 Survey procedure

In order to build our corpus of relevant systems, we performed a comprehensive research in three steps. In the first step, we considered the proceedings of 13 journals, conferences, and workshops (see Table 2). We checked for all papers by inspecting the title and abstract from these proceedings/journals regarding relevant terms such as "programming", "exercise", "automatic", "system", "tutor", "grading", "feedback", or "assessment" (and synonyms). We also consulted the existing reviews outlined in Table 1 as well as the reviews [23, 40] and analyzed whether they report on potentially relevant systems. After that we applied two steps of snowballing.

**Table 2.** Publication venues included in the first round of the literature survey process

| Abbreviation and Title of Journal or Conference (Table of Contents at DBLP or ACM) | Issues Considered in the Survey |
|---|---|
| ABP – German Workshop "Automatische Bewertung von Programmieraufgaben" (Automatic Assessment of Programming Tasks) (https://dblp.org/db/conf/abp/) | completely including 2019 |
| CSEDU – Int. Conf. on Computer Supported Education (https://dblp.org/db/conf/csedu/) | completely including 2020 |
| C&E – Computers & Education Journal (https://dblp.org/db/journals/ce/) | Volume 50 to Volume 159 |
| DELFI – Educational Technologies Conf. of the German Computer Society (https://dblp.org/db/conf/delfi/) | 2008 to 2020 |
| EC-TEL – European Conf. on Technology-Enhanced Learning (https://dblp.org/db/conf/ectel/) | 2008 to 2020 |
| ICALT – Int. Conf. on Advanced Learning Technologies and Technology-enhanced Learning (https://dblp.org/db/conf/icalt/) | 2008 to 2020 |
| ITiCSE – Conf. on Innovation and Technology in Computer Science Education (https://dl.acm.org/event.cfm?id=RE279) | 2008 to 2020 |
| LaTiCE – Learning and Teaching in Computing and Engineering Conf. (https://dblp.org/db/conf/latice/) | complete |
| SIGCSE – Technical Symposium of the Special Interest Group on Computer Science Education (https://dl.acm.org/event.cfm?id=RE175) | 2008 to 2020 |
| TEA/CAA – Technology Enhanced Assessment Conf. (https://dblp.org/db/conf/tea/) | complete |
| TLT – IEEE Transactions on Learning Technologies (https://dblp.org/db/journals/tlt/) | completely including Volume 13(4) |
| TOCE/JERIC – ACM Transact. on Computing Education (https://dblp.org/db/journals/jeric/) | Volume 7(4) to Volume 20(4) |

In the second step, we used Google Scholar and DBLP (author search) to search for additional papers on each system and judged every system based on all available papers according to the inclusion criteria named above and excluded systems if necessary.

In the third step, we inspected the references from the remaining papers similar to the first step. All papers discovered that way were also processed starting from step two until reaching convergency.

We started the search and corpus building in 2018 and finished paper collection for this paper in December 2021. In total, we discovered 572 publications for 298 systems. From those, we excluded 120 systems (210 publications) due to the selection criteria. Finally, 178 systems described by 362 publications are included in the current version of the corpus.

## 4 General characterization

This section provides general information about the systems included in the corpus. We analyzed first usage years and countries of origin as meta-data on the systems, as well as the target groups and self-characterizations of the systems as basic information on the systems' contexts.

### 4.1 First usage years

The first result presented here is an overview of the years in which the systems were used for the first time. In some papers, the first usage year was explicitly stated – if not, the publication year of the oldest paper in the corpus for the specific system was used. The data reveals interesting insights: Approx. 81 % of the systems (145) were first used since 2008. The majority of systems (56 %; 99) seems to be developed and used since 2012 (i.e., median is 2012) whereas approx. 26 % (45) in the last five years since 2016 inclusive. Salient is a peak in 2013 (cf. Figure 1), where 20 systems were used for the first time. Similar peaks for 2008 and 2012 of publications dates could also be observed in an earlier survey [24]. Notably, there are systems that have been developed earlier and only been published before 2008. Since the selection policy excludes these systems, the distribution by years is obviously biased. That is not necessarily a problem, as the corpus is primarily intended for research on the current state of the art and not (yet) for historical research. Due to the large number of systems developed in recent years the contribution to the (technological) state of the art of a system developed and published more than 13 years ago can be considered low.

Nevertheless, seven included systems had their first usage 20 or more years ago. Notable are two systems: TEx-Sys was first used in 1992 (oldest system in our review, [41]) and had its two most recent publications in 2008 [42, 43], whereas ELM-ART was first used in 1992 [44] and was last published in 2016 [45]. So at least the latter seems to be still in use and under investigation nowadays.

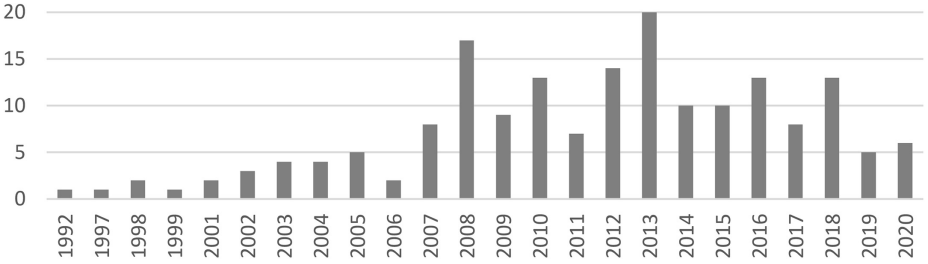


**Fig. 1.** Number of systems regarding their first usage year

### 4.2 Countries

Another aspect we analyzed are the countries from which the systems originate. If the originating country was not explicitly named (which is the case for almost all systems), we looked at the provided postal addresses of the authors of the earliest publication we could find. The countries we found were unambiguous for almost all systems, except for five which have authors from two different countries. For these systems, the country of the first author is used in the analysis.

The majority of systems (90 of 178) have their origin in Europe, followed by North America with 50 systems. In Europe most systems originate from Germany (32) followed by Spain (15). Most systems in North America originate from the USA (40) followed by Canada (7). Hence, about 40 % of the systems come from the USA and Germany. There are 14 systems originating from Asia (particularly, 5 from China as well as 2 from each India and Taiwan), 11 from Oceania (8 from Australia and 3 from New Zealand), and 8 from South America (5 from Brasilia, 2 from Colombia and one from Argentina). Finally, there are only 5 systems from Africa (2 from South Africa, and 1 from each Egypt, Algeria and Morocco).

Cross country cooperation could only be observed for 17 systems. For this we analyzed whether the first paper was a cross country cooperation or whether a follow-up paper was published with one of the original authors and at least one author from a different country. An in-depth investigation showed that there seem to be two (not mutually exclusive) main reasons for a joint publication: (1) A system is used at a different place (3 systems) and (2) a system is jointly developed and researched (7 systems). Unclear regarding these two categories are 6 systems. Apart from joint cross-country publications we found 4 systems for which experience reports were published from distinct authors in different countries.

In general, there seem to be active communities in the USA, Germany and Spain. It would also be interesting to see whether these communities are also visible in their publication (e. g., where is published) and citing (e. g., who is cited) behavior. This requires further investigation.

A reason why Germany is so prominent in this review might be a very active community which the two authors are part of and, therefore, also papers written in German could be included. There are 11 systems which were only published in German. Still, most of the 32 systems originating from Germany have at least one publication in English which makes those accessible to the international research community. These numbers might indicate that many systems get published in English sooner or later. Nevertheless, there is a certain bias in the community analysis, because we were unable to examine other local communities with the same quality. However, there are at least review papers from China [18], India [29], and Brazil [24] which we took into account when surveying systems. Local communities would be expected to be visible there.

### 4.3 Target group

It seems as if almost all systems (168) have been developed for or used in university scenarios, including universities of applied sciences, colleges and similar. However, that does not imply that these systems are solely used or developed for universities. There are four systems which were specially designed for (secondary) schools. However, there are 21 systems for which we found multiple usage scenarios (university and (high) school, university and MOOCS, …). Several systems were used and developed for a target group beyond formal education: Six systems are particularly designed for supporting MOOCs. Different types of contests were mentioned for 9 systems. One commercial system (Automata [46]) focuses on the assessment of programming skills of job applicants.

Generally, the determination of target groups is ambiguous (College = University?) and thus not easy. Oftentimes it is not explicitly stated but can only be inferred using the description of the evaluation (e. g., in a university course).

### 4.4 Self-characterization

As indicated by the analysis above, the corpus contains systems with different goals and approaches. Therefore, it would be interesting to see how the authors characterize, name and describe their systems themselves. A deeper understanding would allow to better find such systems in the future or cluster similar systems. That is particularly important with respect to text and keyword searches that are usually applied when conducting systematic literature reviews. Hence, we analyzed the article keywords and index terms and investigated the system names, the long forms of abbreviations and descriptions which often occur in the form of “Systemname is a …” or “This paper presents a …”. We collected all these descriptions, extracted the main descriptive terms and clustered these.

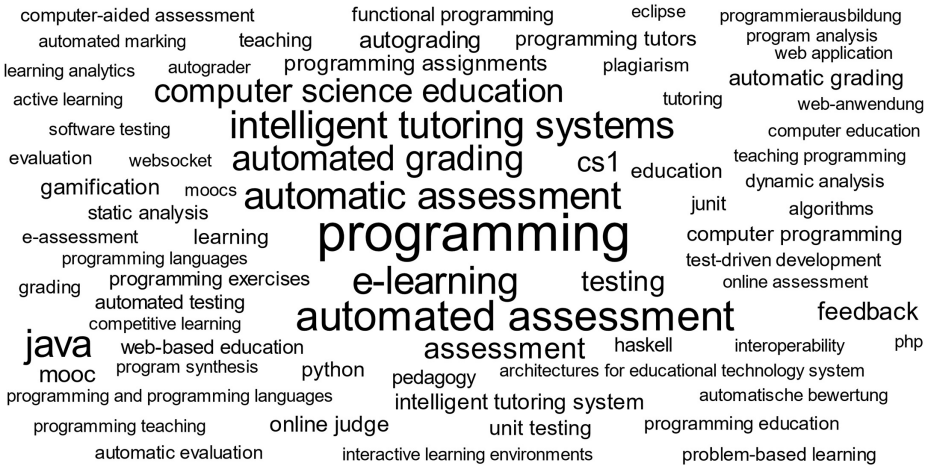


**Fig. 2.** Word cloud of the used index terms/keywords occurring at least three times (71 words)

Figure 2 shows a word cloud of the (unnormalized) keywords and index terms of the articles – the font size corresponds to the frequency of the term. In general, we found 582 terms. The median frequency of the used terms is 1 (139 terms were used at least twice and only 11 terms more than 10 times). The most prominent term is "programming" (31) followed by "automated assessment" (23).

While keywords seem to be chosen quite arbitrarily, descriptions are more structured in most cases. First, there are a few general (not mutually exclusive, often technical) characterizations of the nature of the systems: The most frequent terms are system (89x; this is why we use that term throughout this paper), tool (34x), environment (30x), framework (17x) and application (6x). These terms sometimes seem to be used synonymously. Notable are also more specific terms such as plugin (5x), infrastructure (3x), service (3x), game (2x), (web-based) center (2x), wizard (1x) and wrapper (1x).

Second, many descriptions contain a term characterizing the purpose of the system. The majority of descriptions includes a variation of "assessment/grading/marking". We were able to identify a total amount of 11 different groups of terms that all seem to related directly to summative assessment (cf. Table 3). Apart from summative assessment, frequent terms are "learning" or "tutor(ing)". These characterizations seem to indicate that there are systems which focus on assessing, grading or marking students' work and systems which focus on supporting students with their exercises by giving some kind of feedback without grading. The term feedback itself is used in the descriptions of 16 systems. It appears often as an addition to nature and purpose of the system in the form "… with (automatic) feedback" or alike. Other variations such as "(code) critique" or "hints" are used by two systems and "scaffold" and "visualization" are used by single systems. One subgroup that can be identified quite clearly is a group of systems used in contest scenarios, which are often called "judges" – a term that is used very rarely otherwise. There also seems to be a special subgroup for "hint systems". However, this is not visible solely based on the self-characterization and used keywords but requires a full-text search.

**Table 3.** Overview of used variations of assessment/grading/marking/etc. terms

| Terms | # Sys. | Terms Cont. | # Sys. |
|---|---|---|---|
| assessment/assessing | 47 | judging/judge | 6 |
| grading/grader/… | 33 | exam/examination | 4 |
| test/testing | 13 | validation/validate | 3 |
| marking/marker/mark | 10 | checker/checks | 2 |
| evaluation | 8 | scoring | 1 |
| correction/correctness | 5 | | |

**Table 4.** Overview of used terms for describing the system focus (found at least twice; non-mutually exclusive)

| Terms | # Systems |
|---|---|
| assessment environment\|system\|platform\|framework\|… | 34 |
| tutor(ing) (system\|environment\|framework) | 21 |
| (auto)grader/grading system\|framework\|… | 20 |
| learning (management) environment\|platform\|system | 19 |
| programming\|coding\|development environment\|tool\|… | 11 |
| exercise (management) system\|platform\|framework\|… | 7 |
| judge\|judging system | 7 |
| feedback framework\|(provisioning) system | 6 |
| marker/marking\|scoring environment\|system\|tool | 6 |
| lab(oratory) | 5 |
| teaching tool\|system\|assistant | 5 |
| testing tool\|system\|environment\|framework | 4 |
| contest (management) system | 3 |
| (drill-and-)practice (system) | 3 |
| educational\|serious game/gaming environment | 3 |
| evaluation platform\|service | 3 |
| exam(ination) environment\|system | 3 |
| submission system | 3 |
| ((unit) test) generator\|generation (system) | 3 |
| automation\|compiling system | 2 |

Finally, often adjectives are added to emphasize automation: automatic/automated/… (72x), mechanized (1x), assisted (1x), and in direct combination with the word "grader" as "autograder" (5x). Other significant adjectives are intelligent (17x; always in combination with tutor except for 3 systems (learning/development environment/platform)), interactive (10x), gaming/gamified/game based (5x), adaptive (3x), and virtual (3x in combination with lab and once with each learning environment and examination system).

Table 4 lists the most frequent combination of terms in descriptions. From that list it seems impossible to create simple classifications by a small set of terms. A possible generic classification could be intelligent tutors, exercise-centered systems (with detailed feedback, visualization, and/or hints), as well as contest systems and assessment systems (with a focus on grading/marking/judging). However, this classification is still not partitioning.

Some descriptions also refer to their subject, i. e. the activities or tasks that the students perform. Frequent terms are exercises (24x), assignments (19x), submission (16x), lab (6x), solutions (5x), contest (5x), homework (4x), problem (4x) project (1x), and challenge (1x).

These results indicate that keywords and descriptions are not used consistently within the community. There also seem to be different communities working on support systems for programming with different focuses (e. g., intelligent tutors, assessment/graders, contest judges, and feedback/hint systems). Here a deeper analysis on subgroups in cited articles should be performed.

Please note that this analysis is subjective on three levels: First, these results are based on self-characterization statements that were created by independent, individual authors. The choice of terms does not need to have the very same meaning to different persons, especially (but not only) for non-native English writers and readers. Second, the systems are not static and, thus, might change over time as the development of a system proceeds. Therefore, there might be different statements in different publications which describe a system at a particular point in time – we tried to cover all of these. Third, if there are several statements available for a paper the selection of the terms for analysis might have been arbitrarily chosen. To mitigate these issues the analysis was done carefully by only one single author of this paper who checked all papers individually and tried to honor all relevant statements. Still, we argue that the presented interpretations and the summary of different terms help researchers to connect and allow them to better find relevant papers by knowing keywords they have to look for.

## 5 Discussion & outlook

One of the goals set out for construction of the corpus was to provide maximum coverage of the current state of the art on grading and feedback systems. We cannot guarantee that we have found every relevant system that was published since 2008. Admittedly, that date is quite arbitrary, based on the idea to cover at least 10 years when we started our endeavor of building a corpus in 2018. Independent of that, completeness is a serious issue. The authors of two reviews which include a large number of systems stress that there are no reviews actually striving for completeness [24, 27]. Still, using relevant conferences, journals as a starting point as well as existing review papers, should have allowed us to get a high coverage. If a system was not encountered during our snowball search, this also means that it was not referenced from any paper (e. g., as related work). Our results in Section 4 indicate that a simple keyword search would likely not have brought up all these 178 systems.

The selection of venues might be seen as arbitrary and was solely based on the experience of the authors. There are other possible venues such as ACE, AIED, CSERC,

EDUCON, FiE, ICER, ICSE, IJAIED, ITS and SIGITE which must be considered systematically for future updates of the corpus. Similarly, a deeper analysis of more non-English communities is necessary, as it was done so far only for the German community. Researchers using the current version of the corpus should thus be aware of the filter mechanism to extract publications in a specific language, as the inclusion of articles published in other languages than English could be seen as a threat to validity in some literature studies.

Several papers discovered during the search are vague in some details or allow only indirect conclusions. The authors may have had their personal bias in interpretation of such findings or approaches in relation to the inclusion criteria, which in turn may have influenced the decision on when to include or exclude a system (or consider an improved version with a new name as a completely new system). The same goes for the self-characterizations of the individual systems as already discussed in the previous section. The authors also acknowledge that there is a high number of prominent (commercial) systems that are not considered in this corpus because no scientific publications exist. Finally, new systems are developed and yet unknown publications may provide new facts leading to an inclusion of a currently excluded system. Hence, the work on the corpus is not finished and will likely never be. Consequently, the corpus is published as an interactive research tool rather than a final piece of work. The authors provide and curate the data set in a machine-readable format on https://systemscorpus.strickroth.net. The community is invited to suggest new entries as well as updates to the corpus and to use it as a basis for their own research. There are also methods investigated on how fellow researchers and developers of systems can verify the data in the corpus and correct or extend it (e.g., using GitHub/GitLab merge requests).

Such collaborative work on the corpus is particularly necessary to cover details of the system capabilities. Many details are rarely available in publications in a comprehensive manner due to the publication processes. For example, plain lists of programming languages supported by a particular system often do not add much value to the scientific contribution of a single paper and are thus often omitted. Nevertheless, they are helpful when comparing systems in large scale. Information on supported programming languages are thus included in the corpus but have not been analyzed here due to the assumed incompleteness of the information provided in the papers.

The corpus may promote research in several directions: As already mentioned in Section 2, existing reviews cover between 11 and 101 systems with little overlap on the systems analyzed in these papers. The corpus may facilitate a systematic analysis of a broader view on the existing systems. This could e. g., also reveal (sub)communities and help connecting them (cf. Section 4). Besides the review by [19] which focuses on the interoperability of tools, there is no review analyzing architectures and concrete technical solutions which could provide developers with information about proven approaches or possibilities for reusing existing systems (cf. [27]). The authors are also not aware of reviews on pedagogic, social, and evaluation approaches used in conjunction with such systems. Similarly, security of grading systems is mentioned repeatedly in reviews such as [6, 17] in 2005 and 2010 but there is no review that includes a deeper analysis dedicated to that topic.

## 6 References


[1] R. S. N. Lindberg, T. H. Laine, and L. Haaranen, "Gamifying programming education in k-12: A review of programming curricula in seven countries and programming games," *BJET*, vol. 50, no. 4, pp. 1979–1995, 2019. https://doi.org/10.1111/bjet.12685

[2] A. Luxton-Reilly, Simon, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo, "Introductory programming: A systematic literature review," in *Proc. ITiCSE*, 2018, pp. 55–106. https://doi.org/10.1145/3293881.3295779

[3] D. G. Kay, T. Scott, P. Isaacson, and K. A. Reek, "Automated grading assistance for student programs," *SIGCSE Bull.*, vol. 26, no. 1, pp. 381–382, 1994. https://doi.org/10.1145/191033.191184

[4] K. M. Ala-Mutka, "A survey of automated assessment approaches for programming assignments," *Computer Science Education*, vol. 15, no. 2, pp. 83–102, 2005. https://doi.org/10.1080/08993400500150747

[5] J. Hollingsworth, "Automatic graders for programming classes," *Communications of the ACM*, vol. 3, no. 10, pp. 528–529, Oct. 1960. https://doi.org/10.1145/367415.367422

[6] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä, "Review of recent systems for automatic assessment of programming assignments," in *Proc. Koli Calling*. ACM, 2010, pp. 86–93. https://doi.org/10.1145/1930464.1930480

[7] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, "A survey of literature on the teaching of introductory programming," in *ITiCSE-WGR*, 2007, pp. 204–223. https://doi.org/10.1145/1345375.1345441

[8] R. S. Pettit, J. Homer, and R. Gee, "Do enhanced compiler error messages help students? results inconclusive." in *Proc. SIGCSE*, 2017, pp. 465–470. https://doi.org/10.1145/3017680.3017768

[9] M. Ducassé and A.-M. Emde, "A review of automated debugging systems: Knowledge, strategies and techniques," in *Proc. ICSE*, 1988, pp. 162–171. https://doi.org/10.1109/ICSE.1988.93698

[10] F. P. Deek and J. A. McHugh, "A survey and critical analysis of tools for learning programming," *Computer Science Education*, vol. 8, no. 2, pp. 130–178, 1998. https://doi.org/10.1076/csed.8.2.130.3820

[11] J. P. Leal and N. Moreira, "Automatic grading of programming exercises," DCC-FC & LIACC, Universidade do Porto, Tech. Rep., 1998.

[12] F. Deek, K.-W. Ho, and H. Ramadhan, "A review of web-based learning systems for programming," in *Proc. ED-MEDIA*. AACE, 2001, pp. 382–387.

[13] N. Pillay, "Developing intelligent programming tutors for novice programmers," *SIGCSE Bull.*, vol. 35, no. 2, pp. 78–82, 2003. https://doi.org/10.1145/782941.782986

[14] C. Kelleher and R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," *ACM Computing Surveys*, vol. 37, no. 2, pp. 83–137, 2005. https://doi.org/10.1145/1089733.1089734

[15] M. Guzdial, "Programming environments for novices," in *Computer Science Education Research*. London, UK: Taylor & Francis Group, plc, 2004, pp. 127–154.

[16] M. Gomez-Albarran, "The teaching and learning of programming: A survey of supporting software tools," *The Computer Journal*, vol. 48, no. 2, pp. 131–144, 2005. https://doi.org/10.1093/comjnl/bxh080

[17] C. Douce, D. Livingstone, and J. Orwell, "Automatic test-based assessment of programming: A review," *J. Educ. Resour. Comput.*, vol. 5, no. 3, 2005. https://doi.org/10.1145/1163405.1163409

[18] Y. Liang, Q. Liu, J. Xu, and D. Wang, "The recent development of automated programming assessment," in *Proc. CISE*, Dec. 2009. https://doi.org/10.1109/CISE.2009.5365307

[19] R. Queirós and J. P. Leal, "Programming exercises evaluation systems – an interoperability survey," in *Proc. CSEDU*. SciTePress, 2012, pp. 83–90. https://doi.org/10.5220/0003924900830090

[20] J. Sorva, V. Karavirta, and L. Malmi, "A review of generic program visualization systems for introductory programming education," *TOCE*, vol. 13, no. 4, Nov. 2013. https://doi.org/10.1145/2490822

[21] J. C. Caiza and J. M. del Álamo Ramiro, "Programming assignments automatic grading: Review of tools and implementations," in *Proc. INTED*, 2013, pp. 5691–5700.

[22] N.-T. Le, S. Strickroth, S. Gross, and N. Pinkwart, "A review of AI-supported tutoring approaches for learning programming," in *Proc. ICCSMA*. Springer, 2013, pp. 267–279. https://doi.org/10.1007/978-3-319-00293-4_20

[23] M. Striewe and M. Goedicke, "A review of static analysis approaches for programming exercises," in *Proc. CAA*, no. 439, 2014, pp. 100–113. https://doi.org/10.1007/978-3-319-08657-6_10

[24] D. M. de Souza, K. R. Felizardo, and E. F. Barbosa, "A systematic literature review of assessment tools for programming assignments," in *Proc. CSEET*, Apr. 2016, pp. 147–156. https://doi.org/10.1109/CSEET.2016.48

[25] N.-T. Le, "A classification of adaptive feedback in educational systems for programming," *Systems*, vol. 4, no. 2, 2016. https://doi.org/10.3390/systems4020022

[26] H. Keuning, J. Jeuring, and B. Heeren, "Towards a systematic review of automated feedback generation for programming exercises – extended version," Utrecht University, Tech. Rep. Technical Report UU-CS-2016-001, Mar. 2016. https://doi.org/10.1145/2899415.2899422

[27] H. Keuning, J. Jeuring, and B. Heeren, "A systematic literature review of automated feedback generation for programming exercises," *TOCE*, vol. 19, no. 1, 2018. https://doi.org/10.1145/3231711

[28] Z. Ullah, A. Lajis, M. Jamjoom, A. Altalhi, A. Al-Ghamdi, and F. Saleem, "The effect of automatic assessment on novice programming: Strengths and limitations of existing systems," *Comput. Appl. Eng. Educ.*, vol. 26, no. 6, pp. 2328–2341, May 2018. https://doi.org/10.1002/cae.21974

[29] S. Gupta and A. Gupta, "E-assessment tools for programming languages: A review," in *Proc. ACSIS*, 2019, pp. 65–70. https://doi.org/10.15439/2017KM31

[30] J. McBroom, I. Koprinska, and K. Yacef, "A survey of automated programming hint generation–the hints framework," *arXiv preprint arXiv:1908.11566*, 2019.

[31] S. Strickroth and N. Pinkwart, "Automatisierte Bewertung in der Programmierausbildung – Eine Übersicht," in *Automatisierte Bewertung in der Programmierausbildung*, Digitale Medien in der Hochschullehre. Münster, Germany: Waxmann, 2017, pp. 17–38.

[32] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Technical Report EBSE-2007-01, 2007.

[33] M. Dahm, F. Barnjak, and M. Heilemann, "5Code – Eine integrierte Entwicklungsumgebung für Programmieranfänger," in *Proc. DeLFI*, 2015, pp. 119–130.

[34] B. A. Becker, G. Glanville, R. Iwashima, C. McDonnell, K. Goslin, and C. Mooney, "Effective compiler error message enhancement for novice programming students," *Computer Science Education*, vol. 26, no. 2–3, pp. 148–175, 2016. https://doi.org/10.1080/08993408.2016.1225464

[35] N. Yusof, N. A. M. Zin, and N. S. Adnan, "Java programming assessment tool for assignment module in moodle e-learning system," *Procedia – Social and Behavioral Sciences*, vol. 56, pp. 767–773, 2012. https://doi.org/10.1016/j.sbspro.2012.09.714

[36] D. M. Hoffman, M. Lu, and T. Pelton, "A web-based generation and delivery system for active code reading," in *Proc. SIGCSE*, 2011, pp. 483–488. https://doi.org/10.1145/1953163.1953301

[37] N.-T. Le and W. Menzel, "Using weighted constraints to diagnose errors in logic programming – the case of an ill-defined domain," *IJAIED*, vol. 19, no. 4, pp. 382–400, 2009.
[38] J. Holland, A. Mitrovic, and B. Martin, "J-latte: A constraint-based tutor for java," in *Proc. ICCE*, 2009, pp. 142–146.
[39] J. Spacco, J. Strecker, D. Hovemeyer, and W. Pugh, "Software repository mining with marmoset: An automated programming project snapshot and testing system," in *Proc. MSR*, vol. 30, no. 4. ACM, 2005, pp. 1–5. https://doi.org/10.1145/1082983.1083149
[40] N.-T. Le and N. Pinkwart, "Towards a classification for programming exercises," in *Workshop on AIEDCS*, 2014, pp. 51–60.
[41] T. Dadic, S. Stankov, and M. Rosic, "Prototype model of tutoring system for programming," in *Proc. ITI*, 2006, pp. 41–46. https://doi.org/10.1109/ITI.2006.1708449
[42] T. Dadic, S. Stankov, and M. Rosic, "Meaningful learning in the tutoring system for programming," in *Proc. ITI*, Jun. 2008, pp. 483–488. https://doi.org/10.1109/ITI.2008.4588458
[43] S. Stankov, M. Rosić, B. Žitko, and A. Grubišić, "Tex-sys model for building intelligent tutoring systems," *JCE*, vol. 51, no. 3, pp. 1017–1036, 2008. https://doi.org/10.1016/j.compedu.2007.10.002
[44] G. Weber and M. Specht, "User modeling and adaptive navigation support in WWW-based tutoring systems," in *Proc. UM*, 1997, pp. 289–300. https://doi.org/10.1007/978-3-7091-2670-7_30
[45] G. Weber and P. Brusilovsky, "ELM-ART – an interactive and intelligent web-based electronic textbook," *IJAIED*, vol. 26, no. 1, pp. 72–81, Mar. 2016. https://doi.org/10.1007/s40593-015-0066-8
[46] S. Srikant and V. Aggarwal, "A system to grade computer programming skills using machine learning," in *Proc. SIGKDD*, 2014, pp. 1887–1896. https://doi.org/10.1145/2623330.2623377

## 7 Authors

**Sven Strickroth** is a professor for Technology-Enhanced Learning at the Institute for Informatics of Ludwig-Maximilians-Universität München, Germany (Oettingenstraße 67, 80538 München, Germany). He graduated in computer science at Clausthal University of Technology and received his PhD in Computer Science in 2016 at Humboldt-Universität zu Berlin, Germany. His research interests include E-Assessment, Learning Analytics, and Computer-Supported Collaborative Learning. He is a co-founder of a German workshop series on automated assessment of programming assignments. (email: sven.strickroth@ifi.lmu.de, ORCID: https://orcid.org/0000-0002-9647-300X).

**Michael Striewe** is a research associate at paluno – The Ruhr Institute for Software Technology at the University of Duisburg-Essen, Germany (Gerlingstraße 16, 45127 Essen, Germany). He received a diploma degree in computer science in 2007 from the Technical University Dortmund and a PhD degree in computer science in 2014 from the University of Duisburg-Essen. His research interests combine software engineering and technology-enhanced learning with an emphasis on the design and analysis of computer assisted assessment systems. He is co-founder of a national workshop series on automated assessment of programming assignments and co-editor of a book on the same topic. (email: michael.striewe@paluno.uni-due.de, ORCID: https://orcid.org/0000-0001-8866-6971).

Article submitted 2022-03-29. Resubmitted 2022-05-10. Final acceptance 2022-05-12. Final version published as submitted by the authors.