НАУЧНЫЙ ОБЗОР

Научная статья
УДК 159.99

ЭФФЕКТ СЕРИИ ИЛИ EINSTELLUNG У ПРОГРАММИСТОВ: ОБЗОР НАУЧНЫХ ИССЛЕДОВАНИЙ

Александр Викторович Пруцков[1, 2]
[1] Липецкий государственный педагогический университет имени П. П. Семенова-Тян-Шанского, г. Липецк, Россия, mail@prutzkow.com, https://orcid.org/0000-0002-4110-5269
[2] Рязанский государственный радиотехнический университет имени В. Ф. Уткина, г. Рязань, Россия

Аннотация. Эффект серии – это склонность, которая непосредственно предрасполагает организм к одному типу моторного или сознательного действия. Эффект серии наблюдается у людей различных профессий, в том числе и программистов. Цель работы – расширить знания программистов и преподавателей программирования об эффекте серии, его влиянии на работу программистов посредством обзора научных исследований. Результаты этих научных исследований состояли в следующем. Эффект серии является одной из ловушек мышления программиста. Попав в ловушку, программист применяет не самые лучшие решения. Такие решения встречаются в программах современных студентов. При обучении программированию эффект серии наблюдается у половины студентов. На эффект серии влияет последовательность заданий. Чем больше опыт работы программиста, тем реже наблюдается эффект серии, вплоть до его отсутствия. При обновлении вводных курсов по программированию необходимо учитывать эффект серии. Вместе с ним может наблюдаться функциональная устойчивость, проявления которой часты у студентов. Рассмотренные исследования позволили систематизировать результаты, которые могут быть использованы в дальнейших исследованиях и практической работе, и планы исследований (их этапы, использованные задачи, рекомендации). На основе анализа исследований предлагается снижать влияние эффекта серии за счет расширения знаний программистов о новых подходах к решению задач, новых способах применения существующих решений, современных технологиях программирования, самом эффекте серии, а также расширения общего кругозора программистов, формулирования заданий и их порядка выполнения с учетом знания об эффекте серии, адаптации курсов по программированию в вузах с учетом знания об этом эффекте.

Ключевые слова: эффект серии, Einstellung, психология, программирование, программная инженерия, обучение, психологическое исследование, обзор

Для цитирования

Пруцков А. В. Эффект серии или Einstellung у программистов: обзор научных исследований // Векторы психолого-педагогических исследований. 2025. № 1(06). С. 113–122.

SCIENTIFIC REVIEW

Original article

MENTAL SET OR EINSTELLUNG OF PROGRAMMERS: A SURVEY OF STUDIES

Alexander Viktorovich Prutzkov[1, 2]

[1] Lipetsk State Pedagogical University, Lipetsk, Russia, mail@prutzkow.com, https://orcid.org/0000-0002-4110-5269

[2] Ryazan State Radio Engineering University, Ryazan, Russia

Abstract. The Mental set effect (MS, Einstellung) is a tendency that immediately predisposes an organism to one type of motor or conscious action. MS affects people of various professions, including programmers. The purpose of the study is to expand the knowledge of programmers and programming teachers of MS, its influence on the work of programmers through a survey of scientific studies. The results of these studies were as follows. MS is one of the traps of a programmer's thinking. Having fallen into the trap, a programmer applies not the best solutions. Such solutions are found in the programs of modern students. When learning programming, half of the students exhibit MS. The sequence of tasks affects MS. The more experienced the programmers, the less often they exhibit MS, up to its overcoming. When redesigning introductory programming courses, it is necessary to take MS into account. Together with MS, functional fixity can affect, the manifestations of which are frequent among students. The surveyed studies made it possible to systematize their results, which can be used in further studies and practices, and study plans: their stages, used tasks, recommendations. Based on the analysis of the studies, we propose to reduce the effect of MS by expanding the knowledge of programmers of new approaches to solving problems, new ways of applying existing solutions, modern programming technologies, the MS itself, as well as expanding the general outlook of programmers; formulating tasks and their order of execution taking into account knowledge about MS; adapting programming courses in universities considering knowledge of the effect.

Keywords: mental set, Einstellung, psychology, programming, software engineering, learning, psychological study, survey

For citation

Prutskov, A. V. 2025, ‘The effect of a series or Einstellung on programmers: a review of scientific research’, Vectors of psychological and pedagogical research, iss. 1(06), pp. 113–122.

Введение

Наш мозг ленив. При необходимости мыслительной деятельности мозг «подсовывает» нам уже готовое решение: проверенное опытом, подсказанное интуицией или стереотип. Если бы вы обдумывали каждый перенос ноги при ходьбе, то во что превратилась бы прогулка?! Лень мозга является не его особенностью, а особенностью организма контролировать потребление энергии. Сердце бьется с минимально требуемым ритмом для конкретной ситуации. Мышцы необходимо разминать перед нагрузкой. Энергетические процессы организма в ходе эволюции становились более «экономными» [1, с. 175]. Мозг получает до 2/3 суточного объема глюкозы, потребляемого человеческим организмом [2, с. 29], поэтому энергетическое ограничение его работы оправдано. Ограничивает потребление энергии мозга сам мозг. Мозг – это центральный орган, регулирующий потребление энергии [3, с. 144].

Рассмотрим одно проявление лени (или экономии энергии) мозга – эффект серии (mental set, Einstellung, psychological set [4, с. 96], psychic blindness [5, с. 1277], эффект установки [6, с. 42]). Эффект серии (ЭС, Einstellung) исследован А. Лачинсом. А. Лачинс определяет этот эффект так: «Эффект серии создает автоматическое состояние ума, его слепого склада к задачам; задача рассматривается не с ее собственных качеств, а решается уже проверенным способом» [7, с. 15]. А. Лачинс не является открывателем этого эффекта. В своей работе [7, с. 3] он ссылается на определение термина «Einstellung» в словаре [8, с. 371]: «это склонность, которая непосредственно предрасполагает организм к одному типу моторного или сознательного действия». Эксперимент А. Лачинса кратко описан в одной из его работ [9, с. 23]. История этого исследования приводится в другой работе [10].

Ограничим наше исследование влиянием эффекта серии на программистов.

Цель работы – расширить знания программистов и преподавателей программирования об эффекте серии, его влиянии на работу программистов посредством обзора научных исследований.

Структура исследования

Материалами для анализа послужили научные публикации с результатами исследований, связанных с влиянием эффекта серии на программистов. Методом являлся системный анализ. Далее представлены полученные результаты.

Научные исследования о влиянии эффекта серии на программистов

В программировании существуют следующие ловушки мышления:

1) переоценка упорядоченности: тенденция использовать правила простоты и системности;

2) линейное причинно-следственное мышление: одинаковые следствия вызваны одинаковыми причинами;

3) переоценка подтверждающей информации: повторно возникающие события будут повторяться в будущем;

4) обманчивые ассоциации: ассоциация идей является пространственным элементом мышления, необходимым для обучения; не все ассоциации оказываются адекватными;

5) ограничения производительного мышления [4, с. 96].

Последняя ловушка заключается в том, что для решения задач мы используем не все известные пути, средства и действия, ведущие к решению. Мы ищем решение в предопределенной и ограниченной области. Ограничения связаны с эффектом серии и имеют следующие причины:

– использование прошлого опыт в похожих ситуациях;

– приучение и автоматизация через успешное и повторное выполнение одних и тех же действий;

– функциональная устойчивость назначения средств и подходов [11];

– предположение о существующих правилах и порядке.

Ловушки мышления проявляются в следующих ошибках проектирования программ:

– ненатуральные числа (ловушка 1): исключение из рассмотрения того, что переменные могут принимать отрицательные значения;

– исключительные случаи (ловушки 1, 2 и 5): перебор всех элементов набора вне зависимости от условий задачи;

– ненадежная арифметика (ловушка 1): возникновение ошибок деления на ноль при переполнении значения переменной;

– обманчивые имена (ловушка 4): имена элементов программы, вводящие в заблуждение относительно их назначения;

– неполные условия (ловушка 2): проверка не всех условий при решении задачи (например, отсутствие проверки, могут ли 3 числа быть сторонами треугольника);

– неожиданные значения (ловушка 2): незнание особенностей работы элемента программы (например, в языке программирования Java результат вещественного деления на ноль равен положительной бесконечности и исключение Arithmetic Exception не возникнет [12, с. 182]);

– важные незначительные элементы (ловушка 1): ошибки в программе могут возникать в незначащих частях, которые были написаны недостаточно тщательно;

– обманчивая избыточность (ловушка 2): дублирование фрагмента программы вместо выделения его в подпрограмму;

– функциональная устойчивость (ловушка 5): устойчивость логических выражений при решениях, в том числе в условном операторе.

Некоторые из перечисленных ошибок встречаются в программах современных студентов, проходящих курс «Программирование на Java» [13] в Рязанском государственном радиотехническом университете (РГРТУ).

В списке ошибок не встречается ловушка 3. В отдельной работе [4, с. 96] приводится ошибка с ее проявлением: прекращение дальнейшей отладки программы после нахождения источника ошибки или успешного прохождения 2–3 тестов.

В другой работе [14] исследовалось влияние эффекта серии у 73 студентов, изучающих программирование. Студентам были выданы задания, именуемые Rainfall (X) [15, с. 851], TF (Y) [15, с. 852] и Adding Machine (Z) [16, с. 207]. Задания имеют два одинаковых подхода к решению: ввод и обработка данных в одном цикле или ввод данных в одном цикле, а их обработка в другом цикле. Решения заданий X и Y похожи. К части студентов применялось вмешательство, состоящее в демонстрации видео с предварительным объяснением подходов к решению задачи X. В одной группе из 40 студентов выданные задания имели порядок X, Y и Z, а в другой группе из 33 студентов – Z, X и Y. Была введена следующая шкала проявления эффекта серии:

– полный эффект серии: все 3 задания выполнены одинаково;

– частичный эффект серии: 2 из 3 заданий выполнены одинаково;

– отсутствие эффекта серии: все задания выполнены различно.

Если студенты выполняли три задания неправильно, то этот случай считался отсутствием эффекта серии. Были получены следующие результаты:

– эффект серии у студентов, выполнявших задания по порядку Z, X и Y (среднее число проявлений эффекта серии – 1,8), наблюдался реже, чем у студентов, выполняв-

ших задания по порядку X, Y и Z (среднее число проявлений эффекта серии 1,09). Это объясняется схожестью решений заданий X и Y;

– вмешательство увеличило число проявлений эффекта серии;

– у 32,88 % студентов наблюдался полный эффект серии, у 24,66 % – частичный эффект, у 42,46 % студентов эффект отсутствовал;

– предложены рекомендации по проведению аналогичных исследований.

В публикациях ряда исследователей [9] было изучено влияния эффекта серии на программистов на языке Python. 129 программистам выдавались 4 задания для выполнения в среде разработки, затем проводилось вмешательство и выдавались новые 4 аналогичные задания с измененной формулировкой. Программисты были разделены на три группы. В первой группе вмешательство отсутствовало. Во второй и третьей группах были проведены следующие вмешательства:

– группа 2 – смена цветовой схемы среды разработки с предпочитаемой программистами на противоположную (светлую на темную и наоборот);

– группа 3 – смена цветовой схемы как в группе 2, а также предупреждение и инструкция забыть предыдущее решение.

В задания были введены 4 типа проблем, имеющих эффективные и неэффективные решения:

1) перебор элементов списка с получением их индексов;

2) сравнение типов;

3) суммирование элементов списка;

4) перебор символов строки.

Результатами исследования были следующие.

Приблизительно у 57 (44 %) программистов наблюдался эффект серии; у 41 программиста эффект серии наблюдался при разрешении одной проблемы, у 13 программистов – 2 проблем, у 3 программистов – 3 проблем, 4 проблемы не проявились ни у одного программиста; наибольшее число проявлений эффекта было при разрешении проблемы 2.

Средний опыт работы на языке Python программистов с проявлением эффекта – 25 месяцев, программистов без проявления эффекта – 37 месяцев.

Вмешательства не повлияли на преодоление эффекта серии.

До настоящего времени (по нашему опыту) одной из проблем обучения программистов в вузе является низкая мотивация части студентов к обучению. В работах зарубежных авторов исследовалось влияние изменения вводного курса программирования на мотивацию студентов учиться [17]. Курс изменялся в соответствии со следующими принципами.

1. Программирование – это созидательное действие, а не кодирование. Задача программиста шире, чем преобразование алгоритма в текст программы. Созидательность действий программиста состоит в разработке приложений целиком, интеграции их с уже существующими системами.

2. Программирование требует постоянной практики. Решение задачи (включая понимание существующих решений) представляет собой сложный мыслительный процесс, который требует чередования поощрений и усвоений.

3. Необходимыми элементами программирования являются понимание, изменение и разработка. Существование программных библиотек повышает значимость их системной интеграции в разрабатываемую программу. Понимание текста программы становится не менее важным, чем конструирование приложений из простых составляющих.

4. Умения понимать и изменять программы должны быть достаточными для прохождения учебного модуля.

5. Единственный шаблон, выражающий фундаментальный шаблон программирования, формирует основу всей императивной концепции программирования. Этот шаблон – последовательность трех действий: создание набора данных, заполнение набора данных элементами и перебор элементов для получения производных данных.

6. Необходимо распознавать эффект серии. Исходные примеры должны быть аналогичными тем, что студенты встретят далее.

7. Важные понятия должны рассматриваться более одного раза.

8. Необходимо избегать «математического» программирования. «Математическое» программирование – это написание программ математических вычислений. Современные программы ориентированы на обработку данных.

9. Необходимо избегать переменных, принимающих значения набора данных внутри программного блока.

Курс прошли 79 студентов. Сравнение результатов предварительного анонимного опроса и результатов выполнения 8 заданий показало сохранение мотивации студентов изучать программирование.

Зарубежными авторами исследована функциональная устойчивость у студентов, обучающихся программированию на языке Java [18]. Функциональная устойчивость связана только с функцией объекта, а эффект серии – с решением задачи в целом. Тридцати студентам было рассказано о методах indexOf и startsWith класса String и метода parseInt класса Integer. Затем студентам было предложено выполнить 3 задания по программированию с использованием только этих методов и управляющих структур (линейных, условных и циклических). Если студенты не использовали в задании метод indexOf, то это считалось наличием функциональной устойчивости к этому методу и студент зарабатывал один балл. В результате все студенты получили не менее 1 балла.

Способы снижения влияния эффекта серии

Для снижения влияния эффекта серии на выбор метода решения задач необходимо использовать следующие способы.

1. Знать об этом эффекте и рассказывать о нем другим. Предварительное уведомление о существовании эффекта серии [5] или просьба не использовать примененные ранее решения снижают его влияние [19, 20].

2. Расширять знания о решаемых задачах, методах их решения, связанных с ними темах с помощью книг и бесед с коллегами. Например, задания для исследования взяты из известного и доступного источника [9, с. 26].

3. Разнообразить условия задач, например, изменением регулярных иррелевантных параметров [6], поменять последовательность задач [9, с. 24].

4. В обучении следовать следующим рекомендациям [7, с. 92–93]:

– при изучении способа решения однотипных задач преподаватель должен указывать, что некоторые из этих задач могут иметь другое решение; это можно сделать в виде задания выбора наилучшего решения [21];

– отказаться от требования мгновенных ответов студентов, которые могут закрепить решение задачи;

– в тестах требовать не только ответ, но и способ его получения;

– задания должны предполагать не только воспроизведение способа решения, но и созидательные действия в новой ситуации.

Перечисленные способы могут применяться на инструктажах и семинарах по обсуждению программ [4, с. 100] в компаниях по разработке программного обеспечения.

Выводы

Эффект серии является серьезной проблемой как при обучении программистов, так и в их работе, так как снижает производительность труда. Производительность снижается за счет использования неэффективных способов решения задач (например, по времени выполнения или понимания программы). Неэффективные способы решения рано или поздно придется изменять на эффективные, переписывая программу.

Предлагается ряд приемов для снижения влияния этого эффекта:

– расширение знаний программистов о новых подходах к решению задач, новых способах применения существующих решений, современных технологиях программирования, эффекте серии, а также расширение общего кругозора программистов;

– формулирование заданий и их порядка выполнения с учетом знания об эффекте серии;

– адаптация курсов по программированию в вузах с учетом знания об этом эффекте.

Влияние эффекта можно снижать как отдельно, так и вместе с другими негативными психологическими эффектами (например, функциональной устойчивостью).

Полученные результаты рассмотрения исследований могут быть использованы в последующих научных изысканиях и практической работе. При планировании дальнейших исследований в качестве обязательных элементов предлагаются: их этапы, использованные задачи, рекомендации. В то же время в этих исследованиях не обоснованы некоторые решения, например:

– почему студенты, не выполнившие задания, считались не проявившими эффект серии [14];

– почему метод indexOf класса String языка Java был выбран в качестве инструмента для выявления функциональной устойчивости [18].

Заключение

Проведенный обзор научных исследований будет полезен представителям следующих профессий:

– психологам, исследующим эффект серии или мышление программистов;

– педагогам при постановке новых или совершенствовании существующих курсов по программированию;

– программистам для повышения собственной производительности труда.

Знакомство с психологией программирования можно продолжить при изучении отдельных зарубежных публикаций [22].

Перспективным направлением является установление связи между эффектом серии и контролем потребления энергии мозгом.

Несмотря на важность повышения производительности труда программистов, интерес к их психологии по некоторым направлениям невысок. Так, пик публикаций о мысленных образах программирования пришелся на 1990-е годы [23, 24]. Автор надеется, что эта статья пробудит интерес ученых к междисциплинарному исследованию психологии программистов.

Список источников

1. Wittenberger, C. 1970, ‘The Energetic Economy of the Organism in Animal Evolution’, *Acta Biotheoretica,* vol. 19, iss. 3–4, pp. 171–185.

2. Peters, A. 2011, 'The selfish brain: competition for energy resources', *American Journal of Human Biology,* iss. 23, pp. 29–34, doi: 10.1002/ajhb.21106.

3. Peters, A., Schweiger, U., Pellerin, L., Hubold, C., Oltmanns, K. M., Conrad, M., Schultes, B., Born, J. & Fehma, H. L. 2004, 'The selfish brain: competition for energy resources', *Neuroscience and Biobehavioral Reviews,* vol. 28, iss. 2, pp. 143–180, doi: 10.1016/j.neubiorev.2004.03.002.

4. Grams, T. 1988, 'Thinking Traps in Programming – A Systematic Collection of Examples', *IFAC Proceedings Volumes,* vol. 21, iss. 18, pp. 95–100.

5. Lane, D. M. & Jensen, D. G. 1993, 'Einstellung: Knowledge of the Phenomenon Facilitates Problem Solving', *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* vol. 37, iss. 18, pp. 1277–1280.

6. Тухтиева Н. Х. Влияние типов изменения иррелевантных параметров задач на эффект установки // Вестник СПбГУ. Сер. 12. 2014. Вып. 3. С. 41–48.

7. Luchins, A. S. 1942, 'Mechanization in Problem Solving: The Effect of Einstellung', *Psychological Monographs,* vol. 54, iss. 6. p. 3.

8. Warren, H. C. 1934, 'Dictionary of Psychology', Houghton Mifflin Co, Boston, New York.

9. Sergeyuk, A., Titov, S., Golubev, Ya. & Bryksin, T. 2023, 'Overcoming the Mental Set Effect in Programming Problem Solving', Proceedings of the 34th PPIG, pp. 22–36.

10. Binz, M. & Schulz, E. 2023, 'Reconstructing the Einstellung Effect', Computational Brain & Behavior, vol. 6, iss. 3, pp. 526–542.

11. Duncker, K. 1945, 'On Problem-Solving', *Psychological Monographs,* vol. 58, iss. 5.

12. Пруцков А. В. Тонкости программирования в примерах : учебник. М. : Курс, 2022. 232 с.

13. Пруцков А. В. Преподавание программирования на языке Java в вузе: педагогические аспекты // Гуманитарные исследования Центральной России. 2024. № 4. С. 62–68. DOI: 10.24412/2541-9056-2024-433-62-68.

14. Obispo, J. R. C., Castro, F. E. V. C. G. & Rodrigo, M. M. T. 2018, 'Incidence of Einstellung Effect among Programming Students and its Relationship with Achievement', in *1st Information Computing Education, Cebu City, Philippines, October 4–6, 2018,* Cebu City.

15. Soloway, E. 1986, 'Learning to Program = Learning to Construct Mechanisms and Explanations', *Communications of the ACM,* vol. 29, iss. 9, pp. 850–858.

16. Castro, F. E. V. C. G. & Fisler, K. 2016, 'On the Interplay Between Bottom-Up and Datatype-Driven Program Design', in *47th ACM Technical Symposium on Computing Science Education, February 17, 2016*, pp. 205–210.

17. Jones, M. 2007, 'The Redesign of the Delivery of an Introductory Programming Unit', *Innovation in Teaching and Learning in Information and Computer Sciences,* vol. 6, iss. 4, pp. 169–182, doi: 10.11120/ital.2007.06040169.

18. Talandron-Felipe, M. M. & Bonifacio, K. L. 2019, 'Investigating Functional Fixedness among Novice Student Programmers', in *the 27th International Conference on Computers in Education,* December, 2019.

19. Chrysikou, E. G. & Weisberg, R. W. 2005, 'Following the Wrong Footsteps: Fixation Effects of Pictorial Examples in a Design Problem-Solving Task', *Journal of Experimental Psychology: Learning, Memory, and Cognition,* vol. 31, iss. 5, p. 1134.

20. Tempel, T. & Frings, C. 2019, 'Directed Forgetting in Problem Solving', *Acta Psychologica,* iss. 201, 102955, doi: 10.1016/j.actpsy.2019.102955.

21. Wiese, E. S., Yen, M., Chen, A., Santos, L. A. & Fox, A. 2017, 'Teaching Students to Recognize and Implement Good Coding Style', in *the 4th ACM Conference on Learning @ Scale*, pp. 41–50.

22. Lenberg, P., Feldt, R. &Wallgren, L. G. 2015, 'Behavioral Software Engineering: A Definition and Systematic Literature Review', *Journal of Systems and Software,* iss. 107, pp. 15–37.

23. Bidlake, L., Aubanel, E. & Voyer, D. 2020, 'Systematic Literature Review of Empirical Studies on Mental Representations of Programs', *Journal of Systems and Software,* iss. 165, 110565.

24. Heinonen, A., Lehtelä, B., Hellas, A. & Fagerholm, F. 2023, 'Synthesizing research on programmers' mental models of programs, tasks and concepts – A systematic literature review', *Information and Software Technology,* iss. 164, 107300, doi: 10.1016/j.infsof.2023.107300.

References

1. Wittenberger, C. 1970, 'The Energetic Economy of the Organism in Animal Evolution', *Acta Biotheoretica,* vol. 19, iss. 3–4, pp. 171–185.

2. Peters, A. 2011, 'The selfish brain: competition for energy resources', *American Journal of Human Biology,* iss. 23, pp. 29–34, doi: 10.1002/ajhb.21106.

3. Peters, A., Schweiger, U., Pellerin, L., Hubold, C., Oltmanns, K. M., Conrad, M., Schultes, B., Born, J. & Fehma, H. L. 2004, 'The selfish brain: competition for energy resources', *Neuroscience and Biobehavioral Reviews,* vol. 28, iss. 2, pp. 143–180, doi: 10.1016/j.neubiorev.2004.03.002.

4. Grams, T. 1988, 'Thinking Traps in Programming – A Systematic Collection of Examples', *IFAC Proceedings Volumes,* vol. 21, iss. 18, pp. 95–100.

5. Lane, D. M. & Jensen, D. G. 1993, 'Einstellung: Knowledge of the Phenomenon Facilitates Problem Solving', *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* vol. 37, iss. 18, pp. 1277–1280.

6. Tukhtieva, N. H. 2014, 'The influence of types of changes in irrelevant task parameters on the installation effect', *Bulletin of St. Petersburg State University. Series 12,* iss. 3, pp. 41–48.

7. Luchins, A. S. 1942, 'Mechanization in Problem Solving: The Effect of Einstellung', *Psychological Monographs,* vol. 54, iss. 6.

8. Warren, H. C. 1934, *Dictionary of Psychology,* Houghton Mifflin Co.

9. Sergeyuk, A., Titov, S., Golubev, Ya. & Bryksin, T. 2023, 'Overcoming the Mental Set Effect in Programming Problem Solving', in *Proceedings of the 34th PPIG,* pp. 22–36.

10. Binz, M. & Schulz, E. 2023, 'Reconstructing the Einstellung Effect', Computational Brain & Behavior, vol. 6, iss. 3, pp. 526–542.

11. Duncker, K. 1945, 'On Problem-Solving', *Psychological Monographs,* vol. 58, iss. 5.

12. Prutskov, A. V. 2022, *The subtleties of programming in examples: textbook,* Kurs, Moscow.

13. Prutskov, A. V. 2024, 'Teaching Java programming in higher education institutions: pedagogical aspects', *Humanitarian studies of Central Russia,* iss. 4, pp. 62–68, doi: 10.24412/2541-9056-2024-433-62-68.

14. Obispo, J. R. C., Castro, F. E. V. C. G. & Rodrigo, M. M. T. 2018, 'Incidence of Einstellung Effect among Programming Students and its Relationship with Achievement', in *1st Information Computing Education, Cebu City, Philippines, October 4–6, 2018,* Cebu City.

15. Soloway, E. 1986, 'Learning to Program = Learning to Construct Mechanisms and Explanations', *Communications of the ACM,* vol. 29, iss. 9, pp. 850–858.

16. Castro, F. E. V. C. G. & Fisler, K. 2016, 'On the Interplay Between Bottom-Up and Datatype-Driven Program Design', in *47th ACM Technical Symposium on Computing Science Education, February 17, 2016,* pp. 205–210.

17. Jones, M. 2007, 'The Redesign of the Delivery of an Introductory Programming Unit', Innovation in *Teaching and Learning in Information and Computer Sciences,* vol. 6, iss. 4, pp. 169–182, doi: 10.11120/ital.2007.06040169.

18. Talandron-Felipe, M. M. & Bonifacio, K. L. 2019, 'Investigating Functional Fixedness among Novice Student Programmers', in *The 27th International Conference on Computers in Education, December, 2019.*

19. Chrysikou, E. G. & Weisberg, R. W. 2005, 'Following the Wrong Footsteps: Fixation Effects of Pictorial Examples in a Design Problem-Solving Task', *Journal of Experimental Psychology: Learning, Memory, and Cognition,* vol. 31, iss. 5, p. 1134.

20. Tempel, T. & Frings, C. 2019, 'Directed Forgetting in Problem Solving', *Acta Psychologica,* iss. 201, 102955, doi: 10.1016/j.actpsy.2019.102955.

21. Wiese, E. S., Yen, M., Chen, A., Santos, L. A. & Fox, A. 2017, 'Teaching Students to Recognize and Implement Good Coding Style', in *The 4th ACM Conference on Learning @ Scale,* pp. 41–50.

22. Lenberg, P., Feldt, R. &Wallgren, L. G. 2015, 'Behavioral Software Engineering: A Definition and Systematic Literature Review', *Journal of Systems and Software,* iss. 107, pp. 15–37.

23. Bidlake, L., Aubanel, E. & Voyer, D. 2020, 'Systematic Literature Review of Empirical Studies on Mental Representations of Programs', *Journal of Systems and Software,* iss. 165, 110565.

24. Heinonen, A., Lehtelä, B., Hellas, A. & Fagerholm, F. 2023, 'Synthesizing research on programmers' mental models of programs, tasks and concepts – A systematic literature review', *Information and Software Technology,* iss. 164, 107300, doi: 10.1016/j.infsof.2023.107300.

Информация об авторе

А. В. Пруцков – доктор технических наук, доцент, профессор кафедры информатики, информационных технологий и защиты информации (ЛГПУ), профессор кафедры вычислительной и прикладной математики (РГРТУ).

Information about the author

A. V. Prutzkov – Sc.D (Technical Sciences), assistant professor, professor of the Department of computer science, information technologies, and information security (LSPU); professor of the Department of computational and applied mathematics (RSREU).

Примечание

Содержание статьи соответствует научной специальности 5.3.4. Педагогическая психология, психодиагностика цифровых образовательных сред (психологические науки).

Статья поступила в редакцию 20.12.2024; одобрена после рецензирования 09.01.2025; принята к публикации 13.01.2025.

The article was submitted 20.12.2024; approved after reviewing 09.01.2025; accepted for publication 13.01.2025.