

Gamification for Learning Java Programming: Design and Implementation of Java Realm

Jonathan Tapiwanashe Sithole, Moleboge Emma Nhlapo and Karikoga Norman Gorejena

North-West University, Mmabatho, South Africa

JonathanSithole21@gmail.com

Moleboge.Nhlapo@nwu.ac.za

Koga.Gorejena@nwu.ac.za

Abstract: The emergence of gamification, which is the use of game-design elements and mechanics in non-game contexts, offers a promising alternative to traditional teaching and learning. Teaching and learning Java, one of the most commonly used programming languages, presents a steep learning curve due to its strict syntax and abstract concepts, often leading to cognitive overload and demotivation among novice programmers. Despite Java's enduring importance and numerous attempts at reform, undergraduate students continue to experience high levels of disengagement, difficulty, and limited knowledge retention in programming modules. Conventional teaching and learning methods, such as textbooks and traditional lectures, fail to keep students interested, resulting in frustration and poor learning performance. Gamification can help learners improve knowledge acquisition and problem-solving abilities in the context of programming education. The interactive mode of gamification helps students to understand and retain information. This exploratory study investigates how gamification can be leveraged to address the challenges of learning Java programming, by designing and developing a web-based learning application tailored to the needs of IT students. Adopting a design science research (DSR) methodology, the study followed a rigorous cycle of problem identification, objective definition, design, development, and design validation. Data were collected from 20 participants through quantitative questionnaires and six participants through qualitative focus groups to identify user needs and design requirements. The evaluation confirms that JavaRealm successfully translates user-identified requirements into a functional software artefact aligned with learner preferences. Key findings from this exploratory study indicate strong preference for immediate feedback (95% agreement, 19/20 participants, $M = 4.80$, $SD = 0.52$), professional-grade tools simulating Visual Studio Code, and flexible social features with opt-in leaderboards to accommodate diverse competition preferences. The findings reveal that heterogeneous learner preferences, particularly regarding competitive elements require flexible design rather than uniform implementation. The study develops a practical design blueprint for implementing gamification in complex cognitive domains, demonstrating through user-centred design that effective gamification should prioritise scaffolding, authenticity, structured progression, and user autonomy over simple points and badges. This paper concludes at the design validation phase; summative evaluation measuring actual learning outcomes and competency gains remains essential future work. The blueprint provided herein can guide future large-scale empirical validation of gamification in complex cognitive domains.

Keywords: Gamification, Java programming, Education, Design science research (DSR), Web-based application, Educational technology

1. Introduction

Tertiary institutions worldwide are facing continuous crisis in learning Java programming. Basic Java programming modules experience failure rates over 50% (Watson & Li, 2014) due to cognitive overload, student demotivation and absence of pedagogical support. Learning programming requires a high level of cognitive load (Koudsia & Kirchner, 2024). The syntax, complex abstract concepts such as object-oriented programming, coding, continuous practice to gain programming skills make learning programming difficult and add up to the cognitive overload (Cheng et al, 2022).

The traditional pedagogies of teaching based on lectures have failed to effectively connect theory with practice in programming. Traditional teaching methods that rely on textbooks do not keep students engaged, and knowledge is not well retained, with high failure rates (Romero, Usart & Ott, 2015; Guzdia, 2013). Students express frustration with abstract concepts that have nothing to do with the real-time results, especially for Java, where a great deal of boilerplate is needed to build something tangible before results are visible, which creates an unnecessary cognitive load that foils the learning goals. Bergin & Reilly (2006) identified programming self-efficacy as a critical predictor of success, yet students frequently lack confidence in their programming abilities, creating a self-perpetuating cycle of avoidance and underperformance.

The introduction of game design elements and mechanics in non-game contexts (gamification) has been identified as a potential tool to increase engagement, motivation and learning outcomes in educational environments. Meta-analytic reviews demonstrate positive effects of gamification on student motivation and academic performance across diverse domains (Zhan et al, 2022; Sailer & Homner, 2020). However, the transfer of the gamification concepts to an education in a complex cognitive domain like the education of Java

programming has not been sufficiently investigated, and the few studies that have been conducted so far have not provided both empirical foundations for the users' needs and theoretical foundations for the customisation of the design based on psychological theories like Self-Determination Theory (SDT) or Cognitive Load Theory (CLT).

This exploratory study aims at filling this gap and tackles the issue of how gamification can solve specific problems within Java programming education using a rigorous Design Science Research methodology. This study used design validation techniques: quantitative questionnaires (N = 20) and qualitative focus groups (N = 6) to narrow down the user needs and design requirements of a gamified Java learning system. The requirements were distilled along with educational theory to design and develop a web-based gamified learning application for undergraduate IT students called JavaRealm. This paper illustrates the design validation stage of the DSR cycle, which shows how the designed artefact fulfils the identified needs of the users and preferences of the learners. The primary research questions guiding this work are: (1) What user-identified requirements should guide gamification design in Java programming education? (2) How can diverse student preferences for gamification elements be accommodated within a single system design? (3) What design principles grounded in educational theory can translate user requirements into a functional system architecture?

2. Literature Review

2.1 Gamification in Programming Education

The empirical foundation for gamification in programming education has grown substantially. Sailer et al (2017) examined how specific game design elements enhance motivation by satisfying basic psychological needs for competence, autonomy, and relatedness within Self-Determination Theory framework (Ryan & Deci, 2022). Zhan et al (2022) conducted a systematic review of 21 empirical research studies on the use of gamification in the context of programming and found that gamification positively affected student engagement and academic outcomes, with these findings being consistent across the studies. Barna et al (2018) concluded that the students that participated in gamified programming classes showed more engagement and knowledge retention than the students that did not participate in the gamified classes.

Modern studies focus on the personalisation and learner-centred design. Tondello et al (2019) showed that gamified learning experiences tailored to the learner's preferences have a significantly greater level of engagement than generic implementations. Orji, Tondello and Nacke (2018) suggested some frameworks for tailoring gamification to user types. Most importantly, competition and competition-based leaderboards, which can inspire some learners, can discourage others because of social comparison anxiety experienced by the latter (Hamari, Koivisto & Sarsa, 2014). This tension reflects SDT's distinction between competence and autonomy: leaderboards can support relatedness through social interaction, but risk undermining autonomy for learners who must participate regardless of preference, and competence for those who compare unfavourably. The results indicated that flexible design, which meets a wide range of preferences for competition and collaboration is vital to making learning environments inclusive. Current research projects are concentrating on developing Java programming gamification artefacts. Favretto (2023) developed unit testing gamification module to test the quality and quantity of unit tests produced by students. Juguilon et al (2025) created a cross-platform quiz battle game tool for learning Java which aimed at the identification of algorithms utilised for Code Quest. This research builds on their work.

2.2 Cognitive Load Theory and Java Programming Challenges

Cognitive Load Theory, articulated by Sweller (1988), distinguishes between intrinsic cognitive load, extraneous cognitive load, and germane cognitive load. Java imposes substantial intrinsic load through simultaneous mastery of syntax, algorithmic logic and abstract object-oriented design principles. Critically, Java also imposes substantial extraneous cognitive load. Coding required even for trivial programmes forces students to write substantial code before visible output, creating a cognitive burden unrelated to learning objectives. According to Cheng et al (2022), using digital learning is beneficial for developing students' digital skills, decreasing their anxiety level and managing cognitive overload. Robins, Rountree and Rountree (2003) identified "threshold concepts" which are ideas such as object references, inheritance and polymorphism which function as conceptual bottlenecks. The combination of high intrinsic and high extraneous load leaves minimal working memory capacity for productive learning effort.

Together, this literature indicates that gamification can support engagement while imposing design trade-offs, and that Java's cognitive load can be mitigated through instructional design; JavaRealm, developed in this study, brings these theoretical strands together with empirical user requirements to address this gap.

2.3 Self-Determination Theory and Psychological Needs

Self-Determination Theory provides a psychological framework for understanding motivation through three basic needs: competence (feeling effective and capable), autonomy (self-direction and choice), and relatedness (social connection and belonging) (Ryan & Deci, 2022; Deci & Ryan, 2000). These needs are interwoven in gamified contexts of learning. Points and badges meet competence by demonstrating progress that is seen while competitive leaderboards, come with important design tensions, for some learners, competitive leaderboards serve to provide a sense of relatedness via social interaction, but for others particularly those with below average performance, they can serve to strip autonomy by requiring participation and competence by leading to a publicized failure. Social aspects, collaborative achievement, peer support or aggregate progress visualization, are some of the features of good gamification that must be designed so that they meet the need for relatedness for all students, even those who do not opt for competition.

3. Methodology

3.1 Design Science Research Framework and Design Validation

This research employs Design Science Research, a problem-solving paradigm emphasising creation and evaluation of innovative artefacts to address real-world challenges (Vom Brocke, Hevner & Maedche, 2020; Gregor & Hevner, 2013). DSR was selected because the study's objective was not only to describe learner needs but to design, build and evaluate a functional artefact addressing them, which DSR treats as a legitimate research contribution in its own right. DSR at its core aims to support and enable researcher's cognitive abilities of reasoning, innovation and teamwork when designing and developing creative artifacts (Hevner et al, 2024). The DSR methodology is divided into six steps: Problem Identification, Objectives Definition, Design and Development, Demonstration, Evaluation, and Communication (Vom Brocke, Hevner & Maedche, 2020). The design validation phase is done by organizing questionnaires and focus groups to ascertain if the design meets the identified learners' needs in formative evaluation, which is documented in this paper. This is different from summative evaluation assessment which is used to assess learning outcomes, and actual competence gains.

3.2 Design Validation: Sample and Data Collection

This exploratory study collected data from a purposively-sampled group of 20 quantitative participants and six qualitative participants at a South African university within a context of documented high attrition in introductory programming. Inclusion criteria required participants to be 18+ years, capable of informed consent, enrolled in or having completed Java programming courses, and pursuing IT-related degrees. The sample represented diverse academic maturity: 75% were third-year or postgraduate students, providing valuable retrospective insights into pedagogical needs. Of 20 quantitative participants, 19 provided explicit experience-level self-assessment: Beginner ($n = 7$), Intermediate ($n = 8$), Advanced ($n = 4$). One participant did not provide experience-level categorisation; this participant's data was included in overall descriptive analyses but excluded from subgroup-based comparative analyses to ensure analytical precision. This sample size is suitable for exploratory design validation but is insufficient for generalizable claims about programming education more broadly. Findings should be interpreted as preliminary patterns specific to this context, warranting validation in larger, multi-institutional studies.

The quantitative questionnaire employed eight items using five-point Likert scales, measuring constructs aligned with Self-Determination Theory and Cognitive Load Theory: Engagement, Motivation Consistency, Competition Preference, Collaboration Preference, Immediate Feedback Expectations, Perceived Value of Rewards, Usability Expectations, and Intent to Use. These items were developed specifically for this study. Internal consistency was assessed via Cronbach's alpha coefficients ($\alpha = 0.78$ overall, indicating acceptable reliability for this exploratory instrument). Focus group discussions ($N = 6$, 45–60 minutes each) employed semi-structured protocols addressing critical design requirements, gamification element preferences, social design preferences, and reflections on autonomy and connection. Quantitative data analysis used descriptive statistics (means, standard deviations, percentage agreement) as the primary approach. Non-parametric inferential tests (Kruskal-Wallis H tests and Mann-Whitney U tests) were additionally used, as a secondary and exploratory check, to examine whether responses varied across experience groups (Field, 2024); however, small subgroup sizes ($n = 7$, $n = 8$, $n = 4$) limit statistical power, so findings are interpreted as exploratory patterns requiring larger-sample validation. Qualitative data underwent inductive thematic analysis (Braun & Clarke, 2022), identifying recurring themes without imposing a priori categories. All procedures complied with institutional ethics and data protection regulations, with data pseudonymised and stored in encrypted, regionally-compliant Cloud Firestore instances.

3.3 Technical Architecture and Intended Design Features

JavaRealm was developed using the FERN stack: Firebase, Express, React, Node.js, where React.js was used for the frontend interface, Node.js/Express was used for the application logic and API layer and Google Firebase provides cloud-based identity management and data persistence via Cloud Firestore. This architecture satisfies identified non-functional requirements such as performance, responsiveness, scalability, and accessibility. The system incorporates intended design features, which were designed to address identified user requirements and are informed by Self-Determination Theory (competence, autonomy, relatedness) and Cognitive Load Theory.

The Progressive Code Scaffolding feature is designed to reduce extraneous cognitive load through graduated scaffolding. In early levels, students write only active programming logic (e.g., for-loop implementations) whilst the system automatically injects necessary boilerplate (class declarations, main methods, import statements). As students progress through levels, scaffolding gradually decreases, requiring increasingly complete Java code until advanced levels require fully functional Java classes from scratch. This design is intended to implement Cognitive Load Theory by reducing extraneous load during early learning stages. Code execution occurs within isolated ephemeral Docker containers preventing remote code execution vulnerabilities. The Monaco Editor was selected to provide professional-grade development experience based on user feedback indicating preference for authentic tools.

4. Findings

4.1 Design Validation: User-identified Requirements

Questionnaire analysis identified strong user preference for immediate feedback ($M = 4.80$, $SD = 0.52$, with 19/20 participants, or 95%, agreeing or strongly agreeing that instant code compilation feedback would support engagement). This finding was reinforced in qualitative focus groups, where all six participants spontaneously emphasised immediate feedback as essential to system usability. Engagement ($M = 4.55$, $SD = 0.76$), Perceived Value of Rewards ($M = 4.55$, $SD = 0.69$), and Intent to Use ($M = 4.45$, $SD = 0.69$) also showed strong user preference, indicating broad support for gamified learning approaches. However, competition ($M = 4.10$, $SD = 1.02$) and collaboration ($M = 4.30$, $SD = 1.06$) revealed substantial heterogeneity in user preferences: approximately 15% of quantitative participants rated competition as neutral or undesirable. This heterogeneity in competition preferences, which this study terms the "Competition Paradox" (cf. Hamari, Koivisto & Sarsa, 2014), indicates that mandatory competitive elements would alienate a consistent minority of learners, supporting the need for flexible design accommodating diverse preferences. As a secondary, exploratory check, a Kruskal-Wallis H test suggested that "Motivation Consistency" responses may differ by experience level ($H(2) = 8.18$, $p = .017$), with Beginners reporting higher expected benefit than Advanced participants; given the small subgroup sizes ($n = 7$, $n = 8$, $n = 4$), this preliminary pattern warrants validation in larger studies.

4.2 Qualitative Analysis: Critical Design Requirements

Four dominant themes emerged from qualitative analysis, directly addressing the design requirements needed to address identified learner challenges: firstly "Immediate Feedback as Critical Requirement": All six participants identified feedback latency as a critical barrier, requiring instant code compilation verification with specific line-number error identification. This requirement aligns with Cognitive Load Theory predictions that immediate feedback supports working memory efficiency. Second, "Authenticity over Oversimplification": Participants rejected simplified "toy" interfaces, explicitly requesting professional-grade tools mimicking Visual Studio Code. Simultaneously, participants requested content scaffolding that manages cognitive load through pacing and progressively-decreasing boilerplate injection, not through oversimplification that obscures the reality of Java programming. Third, "Social Flexibility to Address Heterogeneous Preferences": One participant explicitly stated competition motivated them; another expressed fear of "feeling bad if not at the top." This internal diversity mirrored the quantitative Competition Paradox. Participants indicated they needed alternatives to mandatory leaderboards specifically, collaborative achievements and peer code-sharing which satisfy social connection needs without forced comparison. Fourth, "Professional Aesthetic": Participants unanimously rejected "childish" or "toy-like" visual design in favour of professional dark mode with minimal ornamentation, treating the learning system as a professional tool rather than a game.

5. Discussion

The design validation phase confirms that JavaRealm successfully addresses identified user requirements and learner preferences. The strong user preference for immediate feedback ($M = 4.80$, 95% agreement) aligns with

theoretical predictions from Cognitive Load Theory that rapid error feedback supports working memory efficiency in learning. The artefact's containerised, real-time code execution architecture was designed to directly satisfy this requirement. Immediate feedback's importance is well-established in the literature; this study confirms that undergraduate IT students at this institution align with that expectation, not that immediate feedback improves learning outcomes, retention, or programming competency, which was not measured here.

The identification of the "Competition Paradox" reveals critical design considerations. Self-Determination Theory predicts that mandatory competitive leaderboards may threaten competence for lower-performing learners experiencing repeated public failure and autonomy by forcing participation in comparison regardless of preference. This study did not directly measure competence threat or autonomy violation, so SDT predictions remain theoretical interpretations rather than empirically confirmed mechanisms. The design opt-in leaderboards paired with complementary social features are intended to address this heterogeneity through psychological need flexibility. However, whether non-competitive learners actually choose collaborative features and whether these features satisfy their relatedness needs remains an open question.

Regarding cognitive load management: Java's abstract object-oriented programming concepts are substantial and unavoidable. However, extraneous load can potentially be managed through instructional design. JavaRealm's Progressive Code Scaffolding was designed with the intention to reduce extraneous load by allowing early-stage learners to focus cognitive resources on algorithmic logic rather than syntax management, whilst advanced learners gradually assume full code responsibility. This design is theoretically grounded in Cognitive Load Theory's principle of extraneous load reduction through pedagogical design. However, measurement of cognitive load through validated instruments is essential to empirically validate whether scaffolding actually reduces cognitive load in practice.

The professional-grade Monaco Editor and industry-aligned visual design directly addressed qualitative feedback: students prefer authentic professional tools over simplified "toy" interfaces. This finding supports the design principle that sophisticated learners respond positively to being treated as professionals, integrating game mechanics (progression, achievement, optional competition) into professional-grade tools rather than wrapping professional content in superficial "fun" packaging.

This exploratory design validation study demonstrates that rigorous user-centred design, informed by psychological theory (Self-Determination Theory, Cognitive Load Theory) and collected from this specific learner population, can produce artefacts with strong alignment to identified user needs and preferences. However, this paper concludes at the design validation phase. The critical question remains on whether JavaRealm produces measurable improvements in actual learning outcomes, student retention, and programming competency is the subject of essential future summative evaluation work.

6. Design Decisions: Translating Requirements into System Architecture

User-identified requirements were systematically translated into design decisions. In response to the critical requirement for immediate feedback, JavaRealm's backend was architected with containerised, real-time code compilation via Docker containers returning output within milliseconds. In response to authenticity requirements, the Monaco Editor was integrated to provide professional-grade code editing experience mimicking industry-standard development environments. In response to cognitive load management needs identified through qualitative feedback, Progressive Code Scaffolding was incorporated as an intended feature. In response to the Competition Paradox, leaderboards were designed as opt-in features accessible via a separate page, emphasising personal progress ("Your Rank") rather than forcing peer comparison. In response to social connection needs for students opting out of competitive features, the system design incorporates complementary social mechanisms designed to address relatedness needs: collaborative achievements requiring peer engagement, a peer code-sharing portal enabling learning from others' code, and anonymous aggregate progress visualisation (showing "X% of all students completed this level") providing social context without comparative pressure. These complementary social features are designed to ensure that students opting out of competitive leaderboards can still satisfy psychological relatedness needs through alternative mechanisms; whether these mechanisms achieve their intended effect remains to be tested. In response to professional aesthetic preferences, the interface design adopts a dark-mode professional aesthetic with terminal-style iconography and minimal ornamentation, avoiding gamified edutainment aesthetics.

7. Conclusion

This exploratory design validation study demonstrates that systematic, user-centred design informed by educational theory can produce gamified learning systems aligned with learner preferences and identified needs. JavaRealm's design, grounded in design validation data from the sample incorporates design features addressing the requirements identified in this study: immediate code compilation feedback (95% user preference), professional-grade tools, progressive scaffolding intended to manage extraneous cognitive load, and flexible social features designed to accommodate heterogeneous learner preferences regarding competition. The identification of the "Competition Paradox" suggests that effective gamification designs require flexibility to accommodate diverse motivational profiles rather than imposing uniform mechanisms across all learners. This exploratory study suggests preliminary patterns specific to this population and context; validation of these design principles requires large-scale implementation across multiple institutions with rigorous outcome measurement.

7.1 Future Work

This paper concludes at the design validation phase of the DSR cycle. The artefact successfully addresses formatively-identified user requirements; however, the critical question of learning effectiveness remains. Summative evaluation is essential as future work. A rigorous summative evaluation will deploy JavaRealm in live undergraduate Java programming courses with quasi-experimental or randomised controlled designs comparing JavaRealm learners to control groups using conventional instruction. This evaluation must measure Learning Outcomes of Java programming competency using validated instruments), Retention Metrics, Cognitive Load Measures, Psychological Need Satisfaction, Sustained Engagement and Student Perceptions. This longitudinal, multi-institutional evaluation would definitively establish whether the design principles translate into measurable improvements in Java programming education outcomes and student success. Such summative evaluation is the critical next phase required to move beyond design validation toward evidence-based claims about learning effectiveness.

Nevertheless, this formative design validation work contributes significant value: it provides user-centred design guidance grounded in learner feedback and educational theory; it demonstrates rigorous DSR methodology applied to educational technology; it identifies the Competition Paradox as a critical design consideration requiring flexible social feature implementation; and it delivers a functional artefact ready for summative evaluation and large-scale implementation. The blueprint provided herein can guide future implementation and empirical validation of gamification in complex cognitive domains.

Acknowledgements

We acknowledge the school of economic sciences and unit for data science and computing at the NWU.

Ethics Declaration: This study has gone through the university ethical approval process and it was approved as a low-risk study, before any data could be collected. The study ensures the confidentiality and privacy of the participants and complies with applicable privacy laws of the region, from data collection, processing, storage, archival up to disposal. Disposal period is 5 years, whereby data is securely destroyed. Only the research team had access to the data. Data encryption is enforced both in transit (HTTPS/TLS) and at rest (Cloud Firestore's default AES-256 encryption). Necessary permissions, such as informed consent and gatekeeper's permission were obtained before data collection.

AI use Declaration: Some parts of this research work have used Generative Artificial Intelligence. Anthropic Claude 4.5 Sonnet AI tool has been used. AI has been used to improve document structure and organization; language refinement for clarity, academic tone and readability, and technical specification organization.

References

- Barna, B. and Fodor, S. (2018) "An empirical study on the use of gamification on IT courses at higher education", In: M. Auer, D. Guralnick and I. Simonics, eds. *Teaching and learning in a digital world*. Berlin: Springer, pp. 684–692. https://doi.org/10.1007/978-3-319-73210-7_80
- Bergin, S. and Reilly, R. (2006) "Predicting introductory programming performance: A multi-institutional multivariate study", *Computer Science Education*, Vol. 16, No. 4, pp. 303–323.
- Braun, V. and Clarke, V. (2022) *Thematic analysis: A practical guide*, 2nd ed., London: Sage Publications.
- Cheng, Y.P., Shen, P.D., Hung, M.L., Tsai, C.W., Lin, C.H. and Hsu, L.C. (2022) "Applying online content-based knowledge awareness and team learning to develop students' programming skills, reduce their anxiety, and regulate cognitive

- load in a cloud classroom", *Universal Access in the Information Society*, Vol. 21, No. 2, pp. 557-572.
<https://doi.org/10.1007/s10209-020-00789-6>
- Deci, E.L. and Ryan, R.M. (2000) "The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior", *Psychological Inquiry*, Vol. 11, No. 4, pp. 227–268.
- Favretto, M. (2023) "Gamification applied to Java development and testing", Politecnico di Torino, (Masters dissertation).
- Field, A. (2024) *Discovering statistics using IBM SPSS statistics*. 6th ed., London: Sage Publications.
- Gregor, S. and Hevner, A.R. (2013) "Positioning and presenting design science research for maximum impact", *MIS Quarterly*, Vol. 37, No. 2, pp. 337–355. <https://www.jstor.org/stable/43825912>
- Guzdial, M. (2013) "Exploring hypotheses about media computation", In: *Proceedings of the 9th Annual International ACM Conference on International Computing Education Research*. San Diego: ACM, pp. 19–26.
- Hamari, J., Koivisto, J. and Sarsa, H. (2014) "Does gamification work? A literature review of empirical studies on gamification", In: *Proceedings of the 47th Hawaii International Conference on System Sciences*, Waikoloa: IEEE, pp. 3025–3034. <https://doi.org/10.1109/HICSS.2014.377>
- Hevner, A.R., Parsons, J., Brendel, A.B., Lukyanenko, R., Tiefenbeck, V., Tremblay, M.C. and Vom Brocke, J. (2024) "Transparency in design science research", *Decision Support Systems*, Vol. 182, p.114236.
<https://doi.org/10.1016/j.dss.2024.114236>
- Juguilon, J., Ramos, S., Oris, J., Saringan, N.A. and Tamayo, J. (2025) "Code Quest: A Gamification Platform for Aspiring Java Programmers", *The International Journal of Science, Technology and Engineering*, Vol. 1, No.1, pp 22 -31.
- Koudsia, S. and Kirchner, M. (2024) "Reducing cognitive overload for students in higher education: A course design case study", *Journal of Higher Education Theory and Practice*, Vol. 24, No. 10, 97-124.
<https://doi.org/10.33423/jhetp.v24i10.7382>
- Orji, R., Tondello, G.F. and Nacke, L.E. (2018) "Personalizing persuasive strategies in gameful systems to gamification user types", In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Montréal: ACM, pp. 1–14. <https://doi.org/10.1145/3173574.3174009>
- Robins, A., Rountree, J. and Rountree, N. (2003) "Learning and teaching programming: A review and discussion", *Computer Science Education*, Vol. 13, No. 2, pp. 137–172. <https://doi.org/10.1076/csed.13.2.137.14200>
- Romero, M., Usart, M. and Ott, M. (2015) "Can serious games contribute to developing and sustaining 21st century skills?", *Games and Culture*, Vol. 10, No. 2, pp. 148–177. <https://doi.org/10.1177/1555412014548919>
- Ryan, R.M. and Deci, E.L. (2022) "Self-determination theory". In *Encyclopedia of quality of life and well-being research*, (pp. 6229-6235), Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-69909-7_2630-2
- Sailer, M., Hense, J.U., Mayr, S.K. and Mandl, H. (2017) "How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction", *Computers in Human Behavior*, Vol. 69, pp. 371–380. <http://dx.doi.org/10.1016/j.chb.2016.12.033>
- Sailer, M. and Homner, L. (2020) "The gamification of learning: A meta-analysis", *Educational Psychology Review*, Vol. 32, No. 1, pp. 77–112. <https://doi.org/10.1007/s10648-019-09498-w>
- Sweller, J. (1988) "Cognitive load during problem solving: Effects on learning", *Cognitive Science*, Vol. 12, No. 2, pp. 257–285.
- Tondello, G.F., Mora, A., Marczewski, A. and Nacke, L.E. (2019) "Empirical validation of the Gamification User Types Hexad scale in English and Spanish", *International Journal of Human-Computer Studies*, Vol. 127, pp. 95–111.
<https://doi.org/10.1016/j.ijhcs.2018.10.002>
- Vom Brocke, J., Hevner, A. and Maedche, A. (2020) "Introduction to design science research", In *Design science research Cases*, pp. 1-13, Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-46781-4_1
- Watson, C. and Li, F.W. (2014) "Failure rates in introductory programming revisited", In: *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, Montréal: ACM, pp. 39–44.
<http://dx.doi.org/10.1145/2591708.2591749>
- Zhan, Z., He, L., Tong, Y., Liang, X., Guo, S. and Lan, X. (2022) "The effectiveness of gamification in programming education: Evidence from a meta-analysis", *Computers and Education: Artificial Intelligence*, Vol. 3, p. 100096.
<https://doi.org/10.1016/j.caeai.2022.100096>