

mgr inż. Ewelina Melaniuk¹

Akademia Bialska im. Jana Pawła II, Biała Podlaska

https://doi.org/10.29316/9788368103281_8

NOWOCZESNE METODY EDUKACJI INFORMATYCZNEJ W SZKOLNICTWIE WYŻSZYM

MODERN METHODS OF COMPUTER SCIENCE EDUCATION IN HIGHER EDUCATION

Streszczenie

Współczesne szkolnictwo wyższe staje przed wyzwaniem dostosowania metod kształcenia do dynamicznie zmieniających się technologii cyfrowych. Edukacja informatyczna jest jednym z obszarów najbardziej podatnych na transformacje i wymaga zastosowania metod aktywizujących, środowisk projektowych, symulacji, pracy zespołowej oraz narzędzi sztucznej inteligencji. Celem niniejszego rozdziału jest przedstawienie przeglądu nowoczesnych metod dydaktycznych stosowanych w edukacji informatycznej na poziomie akademickim, analiza ich efektywności, a także wskazanie kierunków dalszego rozwoju. W rozdziale omówiono m.in. podejścia konstruktywistyczne, metodę problem-based learning, blended learning, wykorzystanie laboratoriów wirtualnych i chmurowych, platform e-learningowych, gier dydaktycznych oraz narzędzi AI wspierających proces uczenia. Przedstawiono także wyzwania i ograniczenia związane z wprowadzaniem innowacji dydaktycznych.

Słowa kluczowe: edukacja informatyczna, project-based learning, computational thinking, blended learning, sztuczna inteligencja w edukacji, cyfrowe narzędzia edukacyjne, szkolnictwo wyższe, laboratoria wirtualne, nowoczesne metody dydaktyczne

¹ <https://orcid.org/0009-0002-9434-6258>, e-mail: e.melaniuk@dyd.akadmiabialska.pl

Abstract

Contemporary higher education faces the challenge of adapting teaching methods to dynamically changing digital technologies. Computer science education is one of the areas most susceptible to transformation and requires the use of engaging methods, project-based environments, simulations, teamwork, and artificial intelligence tools. The purpose of this article is to provide an overview of modern teaching methods used in computer science education at the academic level, analyze their effectiveness, and identify directions for further development. The chapter discusses, among other things, constructivist approaches, problem-based learning, blended learning, the use of virtual and cloud laboratories, e-learning platforms, educational games, and AI tools supporting the learning process. It also presents the challenges and limitations associated with implementing teaching innovations.

Keywords: computer science education, project-based learning, computational thinking, blended learning, artificial intelligence in education, digital educational tools, higher education, virtual laboratories, modern teaching methods

Wstęp

W ostatnich dekadach, dynamiczny rozwój technologii cyfrowych znacząco wpłynął na sposób funkcjonowania szkolnictwa wyższego, a w szczególności na obszar edukacji informatycznej. Współczesny rynek pracy wymaga od absolwentów kierunków informatycznych nie tylko biegłości technicznej, lecz także umiejętności kreatywnego rozwiązywania problemów, pracy zespołowej, adaptacji do zmieniających się technologii oraz ciągłego uczenia się (Redecker, 2017). Z tego względu tradycyjne podejścia dydaktyczne oparte głównie na wykładach i pracy indywidualnej stają się niewystarczające. Konieczne jest wdrażanie nowoczesnych metod edukacyjnych, które odpowiadają wyzwaniom współczesnego środowiska akademickiego i gospodarczego.

Równolegle rośnie znaczenie uczenia się wspieranego technologią – e-learningu, blended learningu, symulacji komputerowych, wirtualnych laboratoriów czy systemów inteligentnego oceniania. Badania wskazują, że wykorzystanie technologii edukacyjnych zwiększa motywację studentów oraz pozwala na efektywne przyswajanie materiału, szczególnie w dziedzinach wymagających pracy z kodem w środowiskach programistycznych (Pachler

i in., 2010; Dziuban i in., 2018). Ponadto rozwój sztucznej inteligencji otwiera nowe możliwości personalizacji procesu uczenia, automatyzacji oceniania, generowania materiałów dydaktycznych oraz wspierania studentów w pracy projektowej.

W edukacji informatycznej szczególnie istotne będą metody aktywizujące, takie jak Problem-Based Learning (PBL), Project-Based Learning, flipped classroom oraz grywalizacja. Wspomagają one kształtowanie umiejętności praktycznych i analitycznych, kluczowych w zawodach programisty, analityka danych, administratora systemów czy specjalisty ds. cyberbezpieczeństwa (Prince, Felder, 2006). Jak podkreśla Laurillard (2012), efektywne nauczanie informatyki wymaga przede wszystkim interakcji, eksperymentowania oraz możliwości natychmiastowego testowania tworzonych rozwiązań, co nie zawsze jest możliwe w tradycyjnej formie kształcenia.

Dodatkowo, globalizacja i rosnąca mobilność studentów prowadzą do większej różnorodności kompetencyjnej i kulturowej w grupach akademickich. Wymusza to stosowanie metod dydaktycznych elastycznych i skalowalnych, umożliwiających, z jednej strony indywidualizację pracy, a z drugiej prowadzenie zajęć w środowiskach międzynarodowych (Bond i in., 2021). Tym samym szkolnictwo wyższe w obszarze informatyki staje przed koniecznością redefinicji modelu kształcenia, który łączy tradycyjne wartości akademickie z nowoczesnymi narzędziami i metodami.

Celem niniejszego rozdziału jest przedstawienie i analiza najważniejszych współczesnych metod edukacji informatycznej w szkołach wyższych, ze szczególnym uwzględnieniem technologii cyfrowych, metod aktywizujących oraz wykorzystania sztucznej inteligencji w procesie dydaktycznym. Omówione zostały teoretyczne podstawy modernizacji edukacji, praktyczne przykłady oraz wyniki badań naukowych potwierdzających skuteczność i konieczność nowoczesnych rozwiązań.

Tło teoretyczne nowoczesnej edukacji informatycznej

Współczesna edukacja informatyczna na poziomie akademickim opiera się na dynamicznej integracji treści technicznych, metod dydaktycznych oraz technologii wspomagających proces uczenia. Rozwój informatyki

wymaga od studentów zdobycia wiedzy teoretycznej, umiejętności praktycznych i kompetencji społecznych, które pozwalają efektywnie funkcjonować w szybko zmieniającym się środowisku pracy.

Fundament współczesnego kształcenia akademickiego stanowią kompetencje cyfrowe, szczególnie w dziedzinie informatyki. Według ram DIG-COMP 2.3 kompetencje cyfrowe obejmują, m.in. umiejętności techniczne, wyszukiwanie i krytyczną ocenę informacji, komunikację cyfrową, tworzenie treści cyfrowych, bezpieczeństwo cyfrowe oraz rozwiązywanie problemów technologicznych (Vuorikari i in., 2022). W kontekście edukacji informatycznej zakres ten ulega rozszerzeniu o kompetencje specjalistyczne, w tym: programowanie w językach wysokiego i niskiego poziomu, projektowanie i analiza algorytmów, zarządzanie systemami w środowisku chmurowym, cyberbezpieczeństwo i testowanie systemów oraz stosowanie narzędzi sztucznej inteligencji i uczenia maszynowego.

Badania wykazują, że rozwój kompetencji cyfrowych nie jest jedynie kwestią poznania technologii, lecz także kształtowania umiejętności krytycznego myślenia i samodzielnego rozwiązywania problemów (Ferrari, 2013; Redecker, 2017). Kompetencje te są niezbędne, by studenci mogli skutecznie realizować projekty informatyczne, wdrażać innowacyjne rozwiązania w przedsiębiorstwach i odpowiadać na zmieniające się wymagania rynku pracy.

Współczesne programy nauczania informatyki coraz częściej integrują praktyczne zadania z laboratoriami chmurowymi, systemami kontroli wersji, platformami do nauki kodowania oraz narzędziami wspierającymi automatyczne ocenianie (Keuning i in., 2018). Takie podejście pozwala na rozwijanie kompetencji cyfrowych w kontekście realnych problemów technologicznych, co zwiększa skuteczność procesu kształcenia.

Konstruktywizm zakłada, że wiedza powstaje na drodze aktywnego konstruowania osobistych doświadczeń oraz refleksji nad nimi (Piaget, 1970; Vygotsky, 1978). W edukacji informatycznej przekłada się to na: samodzielne projektowanie aplikacji i systemów informatycznych, rozwiązywanie złożonych problemów programistycznych w laboratoriach praktycznych, a także analizę kodu, debugowanie i optymalizację algorytmów.

Konstruktywizm wspiera podejście „learning by doing”, które jest szczególnie istotne w nauce programowania, inżynierii oprogramowania czy administracji systemów. Badania wykazują, że studenci uczestniczący w aktywnych formach nauczania, w tym w projektach zespołowych i eksperymentach praktycznych, osiągają wyższe wyniki i lepiej przyswajają wiedzę (Freeman i in., 2014; Prince, Felder, 2006).

Konektywizm, zaproponowany przez Siemens (2005), traktuje wiedzę jako sieć powiązań między jednostką, społecznością oraz technologią. Uczucie się w kontekście konektywizmu odbywa się poprzez aktywne uczestnictwo w sieciach społecznych, platformach edukacyjnych i repozytoriach wiedzy. W informatyce studenci korzystają, m.in. z:

- GitHub – do zarządzania kodem i współpracy w projektach zespołowych,
- Stack Overflow – do rozwiązywania problemów programistycznych,
- Discord i Slack – do wymiany wiedzy i konsultacji grupowych.

Konektywizm wspiera rozwój umiejętności współpracy, adaptacyjności i krytycznej analizy źródeł informacji, co jest niezbędne w dynamicznie zmieniającym się środowisku IT (Downes, 2007; Kop, Hill, 2008).

Potrzeby rynku pracy

Współczesny rynek pracy w sektorze IT charakteryzuje się dużą dynamiką, zróżnicowaniem specjalizacji oraz szybkim tempem rozwoju technologii. Pracodawcy oczekują od absolwentów solidnych podstaw teoretycznych i umiejętności praktycznych oraz szerokiego wachlarza kompetencji miękkich, umożliwiających efektywne funkcjonowanie w zespołach projektowych, organizacjach międzynarodowych oraz środowiskach pracy zdalnej i hybrydowej (World Economic Forum, 2023). Najważniejsze kompetencje poszukiwane u młodych specjalistów IT obejmują:

- Pracę zespołową i współpracę w środowiskach wirtualnych.

Wielu pracodawców podkreśla, że projekty IT realizowane są obecnie w rozproszonych zespołach, często obejmujących pracowników z różnych krajów i stref czasowych. Oczekuje się więc od kandydatów umiejętności współpracy zdalnej przy wykorzystaniu narzędzi

takich, jak: GitHub, Jira, Figma, Slack czy Microsoft Teams. Efektywna współpraca wymaga także zdolności komunikacyjnych i odpowiedzialności za wykonywane zadania, co jest kluczowe zwłaszcza w projektach realizowanych w metodykach zwinnych.

- Zarządzanie projektami i stosowanie metodyk Agile/DevOps.
Organizacje coraz częściej opierają swoje procesy produkcyjne na metodykach Agile i frameworkach (Scrum czy Kanban). Z kolei podejście DevOps integruje rozwój oprogramowania z procesami wdrożeniowymi, co wymaga zrozumienia automatyzacji testów, CI/CD oraz monitorowania aplikacji (Kim i in., 2016). Studenci informatyki muszą zatem opanować kompetencje techniczne i umiejętności organizacyjne, obejmujące planowanie sprintów, estymację zadań i tworzenie dokumentacji projektowej.
- Umiejętności adaptacyjne i szybkie uczenie się nowych technologii.
Tempo zmian w IT sprawia, że technologie wykorzystywane dziś mogą w krótkim czasie zostać zastąpione nowszymi rozwiązaniami. OECD (2023a) wskazuje, że odporność na zmiany, elastyczność poznawcza oraz gotowość do ciągłego uczenia się są kluczowe w nowych zawodach cyfrowych. Absolwenci powinni być przygotowani do pracy w ekosystemach, w których stale pojawiają się nowe biblioteki, frameworki i narzędzia automatyzacji. Oczekuje się od nich rozumienia podstaw uczenia maszynowego, przetwarzania danych i usług chmurowych, nawet jeśli nie są to ich główne specjalizacje.

Pracodawcy podkreślają, że nawet osoby o wysokich kompetencjach technicznych mogą mieć trudności z jasnym przedstawieniem wyników swojej pracy. W branży IT niezbędne jest przygotowywanie prezentacji dla zespołów projektowych, dokumentacji technicznej, raportów analitycznych i opisów działania systemów. Ponadto komunikacja między działami (np. IT – biznes – UX – QA) wymaga umiejętności tłumaczenia złożonych zagadnień technicznych na język zrozumiały dla odbiorców nietechnicznych (Nakata i in., 2020).

Opisane oczekiwania rynku pracy wskazują, że tradycyjne modele edukacji, oparte na wykładach i zadaniach indywidualnych, nie są wystarczające. Proces kształcenia powinien więc integrować teorię z praktyką poprzez:

- zadania zespołowe i symulacje projektów informatycznych – symulacje typu „mini-software house”, hackathony, realizacja projektów zgodnie z metodyką Scrum pozwalają studentom na rozwijanie kompetencji zgodnych z realnym środowiskiem pracy (Caballé, Conesa, 2018);
- laboratoria wirtualne i środowiska chmurowe – platformy takie, jak AWS Academy, Azure Lab Services, GitHub Classroom umożliwiają studentom pracę w środowiskach identycznych z tymi, które wykorzystuje przemysł IT (Zhong i in., 2020);
- metody aktywizujące: PBL i projekty zespołowe – Problem-Based Learning i Project-Based Learning ułatwiają rozwój kreatywności, myślenia krytycznego oraz kompetencji projektowych. Badania potwierdzają, że studenci informatyki uczący się poprzez projekty osiągają wyższy poziom samodzielności i efektywności w pracy zespołowej (Prince, Felder, 2006; Freeman i in., 2014).

Wdrożenie nowoczesnych metod dydaktycznych umożliwia więc uczelniom przygotowanie absolwentów do rzeczywistych warunków pracy, a studentom – zdobycie doświadczeń zbliżonych do profesjonalnych środowisk projektowych. Jak podkreślają Bond i in. (2021), integracja edukacji technologicznej i praktycznych metod kształcenia jest kluczowym elementem transformacji szkolnictwa wyższego.

Nowoczesne metody dydaktyczne w edukacji informatycznej

Współczesna edukacja informatyczna w szkołach wyższych ulega dynamicznym zmianom pod wpływem rozwoju technologii cyfrowych, zmian na rynku pracy oraz rosnących oczekiwań studentów. Badania wskazują, że tradycyjne wykłady są najmniej efektywną formą nauczania w obszarze informatyki, jeśli nie towarzyszy im komponent aktywnego uczenia się (Freeman i in., 2014). Z tego powodu uczelnie wdrażają metody skoncentrowane na studencie, oparte na rozwiązywaniu problemów, projektach, współpracy

i korzystaniu ze środowisk cyfrowych. W dalszej części rozdziału przedstawiono kluczowe podejścia uznawane za najskuteczniejsze we współczesnej edukacji informatycznej.

Fundament nowoczesnego kształcenia w dyscyplinach STEM stanowi Problem-Based Learning (PBL). Jego głównym założeniem jest uczenie poprzez samodzielne rozwiązywanie problemów, które mają charakter otwarty i często nie posiadają jednoznacznego rozwiązania (Savery, 2015). W kontekście informatyki ten model pozwala studentom rozwijać wiedzę techniczną oraz kluczowe kompetencje XXI wieku: myślenie krytyczne, współpracę, planowanie i analizę. Zastosowanie PBL w edukacji informatycznej obejmuje m.in.: projektowanie systemów bezpieczeństwa informatycznego, analizę złożonych przypadków cyberataków, opracowanie algorytmów optymalizacyjnych oraz diagnostykę błędów w dużych aplikacjach. Badania Prince’a i Feldera (2006) pokazują, że PBL znacząco zwiększa rozumienie materiału i motywację studentów do samodzielnego uczenia się. W kursach programistycznych obserwuje się także poprawę jakości kodu oraz większą świadomość podejmowanych decyzji projektowych.

Naturalnym wyborem w edukacji informatycznej jest Project-Based Learning (PjBL), ponieważ model pracy projektowej odzwierciedla realne praktyki przemysłu IT. Thomas (2000) wykazał, że PBL poprawia motywację, zaangażowanie oraz poczucie odpowiedzialności studentów, a także pozwala im nabyć umiejętności praktyczne potrzebne na rynku pracy. Typowe formy projektów akademickich obejmują: tworzenie aplikacji webowych lub mobilnych, projektowanie systemów IoT, implementację modułów sztucznej inteligencji, analizę danych i przygotowanie modeli predykcyjnych oraz budowę systemów sieciowych i administrację środowiskiem serwerowym. W badaniach przeprowadzonych przez Behera i Dash (2016) wskazano, że praca zespołowa w modelu projektowym zwiększa kompetencje komunikacyjne i poprawia zdolność studentów do pełnienia ról zawodowych: programisty, analityka, testera czy inżyniera DevOps.

Dominującym trendem w europejskich uczelniach stało się Blended Learning, czyli nauczanie hybrydowe łączące tradycyjne zajęcia kontaktowe z e-learningiem (Graham, 2019). Dzięki elastyczności i dostępowi do

zasobów cyfrowych studenci mogą rozwijać kompetencje informatyczne w sposób ciągły i autonomiczny.

Szczególną formą blended learningu jest flipped classroom (odwrócona klasa). W modelu tym studenci przyswajają podstawową wiedzę przed zajęciami (np. z materiałów wideo), zaś na zajęciach koncentrują się na rozwiązywaniu zadań i projektów. Badania Bishop i Verleger (2013) wykazały, że odwrócona klasa sprzyja zwiększonej aktywności studentów, lepszemu rozumieniu materiału oraz większemu zaangażowaniu w zajęcia praktyczne. W informatyce flipped classroom znajduje szczególne zastosowanie w kursach programowania, gdzie część teoretyczną można przekazać w formie nagrań, a cenny czas na zajęciach przeznaczyć na kodowanie, debugging i konsultacje z prowadzącym.

Przełom w edukacji informatycznej stanowi wprowadzenie laboratoriów chmurowych. Tradycyjne laboratoria komputerowe mają wiele ograniczeń, takich jak koszty sprzętowe, trudności w utrzymaniu infrastruktury i ograniczona skalowalność. Wirtualizacja i technologia cloud computing rozwiązują te problemy, umożliwiając studentom dostęp do zaawansowanych narzędzi niezależnie od czasu i miejsca. Laboratoria chmurowe umożliwiają pracę z: klastrami Kubernetes, serwerami wirtualnymi, konteneryzacją aplikacji, wirtualnymi sieciami korporacyjnymi, narzędziami DevOps (Jenkins, GitLab CI, Terraform) oraz platformami big data (Hadoop, Spark). Zhang i in. (2021) podkreślają, że użycie środowisk chmurowych zwiększa efektywność uczenia się administracji systemami i technologii sieciowych nawet o 40%. Dodatkowo uczelnie mogą symulować środowiska produkcyjne, ułatwiając przejście studentów do pracy zawodowej.

Istotnym elementem nowoczesnej edukacji stała się grywalizacja, a w informatyce znajduje szczególne zastosowanie ze względu na naturalną skłonność studentów do rozwiązywania problemów w środowiskach interaktywnych. Deterding i in. (2011) definiują grywalizację jako stosowanie elementów gier w kontekstach niezwiązanych z grami, co umożliwia wzrost motywacji i zaangażowania. Przykłady użycia grywalizacji w edukacji IT to m.in.: konkursy programistyczne (np. hackathony), systemy rankingowe na platformach e-learningowych, interaktywne zadania w środowiskach typu Codewars, symulacje cyberataków (np. capture-the-flag w cyberbezpieczeństwie).

Badania Papastergiou (2009) pokazują, że elementy gier poprawiają wyniki nauczania w obszarach technicznych, zwłaszcza gdy zadania mają charakter rywalizacji i natychmiastowego feedbacku.

Coraz większą popularność w edukacji zyskują techniki Pair Programming i Code Review, które wywodzą się z metodologii Extreme Programming. Williams i Kessler (2003) udowodnili, że programowanie w parach: skraca czas potrzebny na naukę nowych konceptów, zmniejsza liczbę błędów w kodzie, zwiększa satysfakcję studentów oraz poprawia komunikację zespołową. Z kolei Code Review umożliwia studentom nabywanie praktyk stosowanych na co dzień w branży IT – od analizowania stylu kodowania po wykrywanie luk bezpieczeństwa. Tego typu praktyki przyczyniają się do rozwijania kompetencji społecznych, kultury technicznej i odpowiedzialności za jakość pracy.

Sposób nauczania programowania, baz danych czy analizy danych radykalnie zmieniła sztuczna inteligencja (AI). Narzędzia takie, jak GitHub Copilot, ChatGPT, Amazon CodeWhisperer czy systemy automatycznego testowania kodu stają się integralną częścią środowiska pracy programisty i coraz częściej są włączane do procesów dydaktycznych. Badania Denny i in. (2023) wskazują, że korzystanie z asystentów AI może zwiększyć szybkość nauki programowania nawet o 30%, szczególnie u studentów początkujących. Jednak autorzy podkreślają również konieczność zachowania umiaru – nadmierne poleganie na generowanym kodzie może prowadzić do spadku umiejętności samodzielnego rozwiązywania problemów. Najważniejsze korzyści z zastosowania AI to przede wszystkim: możliwość natychmiastowego feedbacku, wspieranie studentów w pisaniu kodu, analiza błędów i ich automatyczne naprawianie, rekomendacje dotyczące stylu i architektury, a także indywidualizacja procesu uczenia się. W tym obszarze szczególnie istotne jest wprowadzenie do programów studiów treści dotyczących etyki AI, odpowiedzialnego wykorzystywania narzędzi oraz analizy ryzyk związanych z prywatnością, błędami algorytmicznymi i jakością kodu (Floridi, Cows, 2019).

Narzędzia wspierające nowoczesną edukację informatyczną

Współczesna edukacja informatyczna opiera się na nowoczesnych metodach dydaktycznych oraz wyspecjalizowanych narzędziach, które usprawniają nauczanie, ocenianie i rozwój kompetencji cyfrowych studentów. Oprogramowanie edukacyjne stosowane w szkolnictwie wyższym pełni wiele funkcji: automatyzuje zadania wykładowcy, zapewnia standaryzację środowiska pracy, wspomaga analizę wyników studentów i daje dostęp do zaawansowanych zasobów obliczeniowych. W tej części rozdziału przedstawiono najważniejsze kategorie narzędzi wykorzystywanych w nowoczesnej edukacji informatycznej.

Fundament infrastruktury dydaktycznej współczesnych uczelni stanowią platformy e-learningowe typu LMS (Learning Management System). Systemy takie, jak Moodle, Blackboard Learn czy Canvas LMS umożliwiają zarządzanie treściami kursów, komunikację ze studentami, przeprowadzanie testów oraz monitorowanie postępów uczestników. Jak wskazują Watson i Watson (2012), platformy LMS odgrywają kluczową rolę w standaryzacji procesu dydaktycznego, a ich rozwój coraz częściej koncentruje się na automatyzacji interakcji i integracji z narzędziami analitycznymi. W edukacji informatycznej LMS pełnią dodatkowe funkcje takie, jak: automatyczne testowanie wiedzy z zakresu programowania, algorytmów, sieci komputerowych, obsługa quizów adaptacyjnych, które dostosowują poziom trudności do umiejętności studenta, integracja z systemami chmurowymi, np. Google Cloud, AWS Educate czy Azure for Students, analiza uczenia się (learning analytics) pozwalająca wykrywać trudności studentów we wczesnym etapie kursu oraz śledzenie aktywności i progresu, co pozwala wykładowcom na projektowanie bardziej spersonalizowanych ścieżek edukacyjnych. W badaniach Hrastinskiego (2019) podkreśla się rosnące znaczenie analityki uczenia opartej na danych, która wspiera podejmowanie decyzji dydaktycznych oraz rozwój funkcji predykcyjnych pozwalających przewidzieć ryzyko niezaliczenia kursu. Dzięki temu platformy e-learningowe stają się narzędziem dystrybucji treści i kluczowym elementem systemowego wsparcia edukacji informatycznej.

Tradycyjne nauczanie programowania wymagało instalacji i konfiguracji lokalnego oprogramowania, co prowadziło do znacznych problemów logistycznych – różnic w systemach operacyjnych, błędów konfiguracji czy braku kompatybilności bibliotek. Środowiska programistyczne online, takie jak GitPod, Replit, CodeSandbox, Google Colab czy AWS Cloud9, eliminują te problemy, dostarczając natychmiast gotowe do pracy, zunifikowane środowiska. Ich kluczowe zalety obejmują: dostęp z dowolnego urządzenia, również słabszego sprzętowo, automatyczne konfigurowanie środowisk, z instalacją bibliotek, frameworków i narzędzi, możliwość pracy grupowej online, w tym edycję kodu w czasie rzeczywistym, integrację z GitHubem i GitLabem, co wprowadza studentów w praktyki DevOps, a także obsługę kontenerów (np. Docker), które ułatwiają naukę mikroservisów i wdrażania aplikacji. W badaniach Marín i in. (2021) wykazano, że wykorzystanie zdalnych środowisk programistycznych zwiększa efektywność nauki programowania o ok. 25-30%, głównie dzięki redukcji problemów technicznych oraz możliwości szybkiego przejścia do właściwej pracy nad kodem. Ponadto ułatwiają one organizację zajęć w modelu hybrydowym i zdalnym, co okazało się szczególnie istotne w okresie pandemii COVID-19.

Środowiska online odgrywają również dużą rolę w kursach związanych z: analizą danych (Python, R), algorytmiką i strukturami danych, programowaniem aplikacji webowych (Node.js, React, Django) i sztuczną inteligencją (TensorFlow, PyTorch). Standardyzacja środowiska pracy znacząco ułatwia ocenianie projektów oraz analizę błędów oraz umożliwia studentom rozwijanie własnych projektów poza zajęciami.

Coraz większą rolę w edukacji informatycznej, szczególnie na dużych kierunkach, gdzie liczba studentów może utrudniać tradycyjne ocenianie, odgrywają automatyczne systemy oceny kodu. Narzędzia takie, jak CodeGrade, HackerRank for Education, Moss, Gradescope, CodeRunner czy Code.org Assessments umożliwiają: automatyczne sprawdzanie poprawności kodu, ocenianie wydajności i złożoności algorytmów, analizę stylu i jakości kodu, detekcję plagiatów programistycznych oraz generowanie natychmiastowej informacji zwrotnej. Keuning, Jeuring i Heeren (2018) dokonali systematycznego przeglądu narzędzi do automatycznego oceniania i wskazali, że redukują one obciążenie wykładowców nawet o 60-70% oraz poprawiają

spójność i obiektywność ocen. Automatyczne testy umożliwiają studentom wielokrotne podejście do zadania, co ma pozytywny wpływ na proces uczenia się, zgodnie z teorią „natychmiastowego feedbacku” Shute (2008).

Automatyzacja oceniania jest szczególnie cenna w kursach: programowania proceduralnego i obiektowego, struktur danych i algorytmów, systemów operacyjnych, baz danych i cyberbezpieczeństwa. Przykładowo w kursach SQL można testować zapytania na dużych zbiorach danych, a w kursach cyberbezpieczeństwa automatycznie oceniać konfiguracje serwerów czy symulacje ataków. Co istotne, narzędzia te mogą być integrowane z LMS, co pozwala uczelniom na tworzenie kompleksowych ekosystemów edukacyjnych obejmujących dystrybucję treści, środowisko pracy oraz ocenę wyników.

Sztuczna inteligencja (AI) odgrywa coraz większą rolę w procesie kształcenia na poziomie akademickim, a jej zastosowanie w edukacji informatycznej staje się jednym z najbardziej dynamicznie rozwijających się obszarów. Narzędzia oparte na AI wspierają prowadzących zajęcia i studentów, umożliwiając automatyzację procesów, personalizację nauczania, symulacje zaawansowanych środowisk oraz wsparcie w analizie kodu. Rozwój technologii dużych modeli językowych (LLM), systemów rekomendacyjnych oraz inteligentnych środowisk programistycznych zmienia sposób uczenia się i nauczania informatyki.

Jednym z kluczowych obszarów zastosowania AI są inteligentne asystenty edukacyjne, takie, jak ChatGPT, Google Gemini, Github Copilot, Codeium, czy narzędzia wbudowane w IDE, np. AI Assistant w IntelliJ IDEA. Systemy te umożliwiają: generowanie przykładowych rozwiązań programistycznych, wyjaśnianie złożonych zagadnień teoretycznych, analizę błędów w kodzie i proponowanie ich poprawy, symulowanie rozmowy z ekspertem lub tutorowanie krok po kroku oraz wspieranie w generowaniu dokumentacji. Badania Denny i in. (2023) wskazują, że studenci korzystający z asystentów AI uczą się programowania szybciej i popełniają mniej błędów syntaktycznych, co przekłada się na wzrost efektywności nawet o 30%. Jednocześnie istotne jest świadome korzystanie z takich narzędzi, ponieważ mogą one generować kod nieoptymalny lub błędny, jeśli problem jest nieprecyzyjnie opisany.

AI wspiera również proces indywidualizacji kształcenia poprzez analizę postępów studenta, identyfikację braków wiedzy oraz rekomendowanie odpowiednich materiałów dydaktycznych. Wykorzystuje się do tego: systemy adaptacyjnego uczenia, inteligentne moduły rekomendacyjne, analizę danych z LMS i detekcję obszarów trudności. Według Chen i in. (2020) narzędzia oparte na AI pozwalają na zwiększenie skuteczności nauki szczególnie u studentów, którzy mają trudności z przyswojeniem bardziej abstrakcyjnych konceptów informatycznych takich, jak złożoność obliczeniowa czy rekurencja. Systemy te umożliwiają: automatyczne dostosowanie poziomu trudności zadań, identyfikację wzorców błędów programistycznych, rekomendowanie dodatkowych ćwiczeń i materiałów, tworzenie profili kompetencyjnych studentów. Personalizacja nauki wpisuje się w paradygmat „learning analytics” i „data-driven education”, szczególnie podkreślany przez Zawackiego-Richtera i współautorów (2019).

AI wspiera także tworzenie zadań dydaktycznych. Narzędzia oparte na modelach językowych potrafią: generować zadania programistyczne na różnym poziomie trudności, tworzyć testy jednostkowe do automatycznego sprawdzania kodu, analizować styl i logikę programowania, wykrywać podobieństwa i potencjalną plagiaty w kodzie. Rozwiązania te są szeroko wykorzystywane w narzędziach takich, jak: CodeGrade AI, HackerRank AI Insights, Codex-based problem generators, MOSS (Machine Learning for plagiarism detection). Badania Keuning i in. (2018) podkreślają, że automatyczne systemy oceny wzbogacone o moduły AI zapewniają bardziej szczegółowy feedback oraz są w stanie wykrywać błędy logiczne, które wcześniej wymagały ręcznego sprawdzania.

Narzędzia AI coraz częściej stanowią integralną część nowoczesnych IDE. Rozwiązania takie, jak: Visual Studio Code (Copilot), JetBrains AI Assistant, Amazon CodeWhisperer, umożliwiają studentom uczenie się poprzez praktykę w środowisku zbliżonym do tego, którego używają profesjonaliści. Inteligentne środowiska potrafią bowiem: przewidywać fragmenty kodu (code completion), analizować jakość projektową (clean code), automatycznie refaktoryzować programy, wykrywać podatności bezpieczeństwa i sugerować optymalne struktury danych. Według Vaithilingam i in. (2022), użycie narzędzi AI w IDE poprawia szybkość pracy studentów o 15-40%,

szczególnie w zadaniach wymagających klejenia kodu lub stosowania znanych wzorców projektowych.

Chatboty oparte na AI pełnią funkcję wirtualnych tutorów dostępnych 24/7. Mogą odpowiadać na pytania studentów, wyjaśniać złożone zagadnienia i pomagać w rozwiązywaniu problemów technicznych. Przykładami takich rozwiązań są: Jill Watson – chatbot wykorzystywany na Georgia Tech (Goel, Polepeddi, 2016), chatboty wdrażane w systemach LMS do obsługi kursów programistycznych, a także narzędzia generatywne typu LLM służące jako mentorzy kodowania. Badania odnotowują, że chatboty edukacyjne mogą zmniejszyć liczbę pytań kierowanych bezpośrednio do wykładowcy o nawet 40-50%, co usprawnia zarządzanie dużymi grupami studentów (Winkler, Söllner, 2018).

W edukacji związanej z bezpieczeństwem IT narzędzia AI wspierają: symulacje cyberataków, analizę logów systemowych, wykrywanie anomalii sieciowych, automatyczne generowanie scenariuszy szkoleniowych. Dzięki temu studenci mogą ćwiczyć praktyczne umiejętności w środowiskach zbliżonych do realnych incydentów (Wada i in., 2025).

Wraz z rosnącą popularnością narzędzi AI pojawia się konieczność rozwijania kompetencji etycznych. Studenci powinni rozumieć ryzyka związane z błędami generowanymi przez modele językowe, problemy uprzedzeń i halucynacji w AI, zagadnienia prawne – w tym prawa autorskie dotyczące kodu generowanego przez AI, granice korzystania z AI podczas egzaminów i projektów oraz konieczność weryfikacji treści i kodu wygenerowanego automatycznie. Edukacja w obszarze AI musi obejmować rozwój umiejętności technicznych i etycznych, aby korzystanie z narzędzi było odpowiedzialne i wspierało rozwój kompetencji studenta zamiast ich zastępowania.

Wyzwania i ograniczenia

Mimo dynamicznego rozwoju narzędzi cyfrowych w edukacji informatycznej, wdrażanie nowoczesnych metod i technologii napotyka na szereg barier i wyzwań. Obejmują one kwestie infrastrukturalne, kompetencyjne, etyczne oraz organizacyjne. Analiza tych obszarów jest kluczowa dla efektywnego planowania transformacji dydaktyki na poziomie akademickim i szkolnym (Selwyn, 2016).

Wiele uczelni nadal boryka się z ograniczeniami infrastrukturalnymi, które utrudniają realizację nowoczesnych form kształcenia. Do najczęściej wskazywanych problemów należą: przestarzały sprzęt komputerowy, niedostateczna przepustowość sieci, ograniczony dostęp do chmury publicznej oraz brak odpowiednich licencji na oprogramowanie specjalistyczne. Badania wykazują, że infrastruktura ICT stanowi jeden z kluczowych czynników warunkujących sukces cyfrowej transformacji uczelni (Kirkwood, Price, 2014). Niewystarczające zaplecze techniczne ogranicza możliwość prowadzenia kursów z wykorzystaniem środowisk wirtualnych, narzędzi AI oraz projektów zespołowych w chmurze.

Również cyberbezpieczeństwo stanowi istotne wyzwanie. Rozbudowa infrastruktury cyfrowej wiąże się z koniecznością stosowania bardziej zaawansowanych procedur ochrony przed atakami – według raportów EDUCAUSE, ponad 60% instytucji szkolnictwa wyższego ocenia swoje zabezpieczenia jako niewystarczające (EDUCAUSE, 2022).

Transformacja edukacji informatycznej wymaga odpowiednio przygotowanej kadry dydaktycznej. Obejmuje to: znajomość metod aktywizujących (PBL, projekty, flipped classroom), kompetencje w zakresie narzędzi chmurowych i automatyzacji, umiejętność wykorzystania narzędzi AI w dydaktyce, a także gotowość do pracy zespołowej i prowadzenia mentoringu. Wielu nauczycieli akademickich deklaruje braki w przygotowaniu do pracy z narzędziami AI oraz środowiskami programistycznymi online. Badania potwierdzają, że kompetencje cyfrowe wykładowców są jedną z głównych barier skutecznego wdrażania innowacji technologicznych. Co więcej, część kadry wykazuje opór przed zmianami wynikający z przywiązania do tradycyjnych metod, braku czasu na podnoszenie kwalifikacji czy obaw przed utratą kontroli nad procesem dydaktycznym.

Dynamiczny rozwój modeli generatywnych (LLM) stwarza także nowe wyzwania dla edukacji informatycznej. Narzędzia AI mogą prowadzić do nadmiernego polegania na automatycznie generowanych odpowiedziach, co osłabia rozwój kompetencji problem-solving i umiejętności programistycznych. Modele generatywne potrafią wytwarzać kod, eseje i rozwiązania zadań trudne do wykrycia tradycyjnymi metodami antyplagiatowymi. Już dziś wiele uczelni raportuje wzrost przypadków nieuczciwości akademickiej

związanej z AI (Cotton i in., 2023). Kolejnym jest problem tzw. „hallucinations” – modele językowe mogą generować błędne lub fikcyjne informacje, co stanowi realne zagrożenie podczas nauki algorytmów, zasad bezpieczeństwa, procedur administracji systemami i praktyk projektowania oprogramowania. W związku z tym konieczne jest uczenie studentów kompetencji krytycznej oceny treści generowanych automatycznie.

Wdrażanie nowoczesnych metod dydaktycznych wiąże się również z szeregiem trudności o charakterze organizacyjnym. Pierwszą niedogodnością jest zwiększony nakład pracy, bowiem metody takie jak PBL czy nauczanie projektowe wymagają przygotowania realnych studiów przypadków, indywidualnej opieki nad zespołami studenckimi, częstszej ewaluacji postępów i czasu na analizę kodu i dokumentacji projektowej. Badania wskazują, że nauczyciele poświęcają nawet 50% więcej czasu na przygotowanie zajęć opartych na metodach aktywizujących w porównaniu z tradycyjnym wykładem (Hmelo-Silver, 2004). Kolejną trudnością jest złożoność logistyki pracy zespołowej – organizacja kursów projektowych wymaga harmonogramowania spotkań, zarządzania repozytoriami kodu, kontroli wersji i stałego monitorowania aktywności zespołów. Dodatkowo, ocenianie projektów aplikacyjnych, prac zespołowych czy zadań tworzonych z pomocą AI wymaga tworzenia bardziej elastycznych, wieloskładnikowych kryteriów, co bywa wyzwaniem w dużych grupach.

Kierunki rozwoju edukacji informatycznej

Edukacja informatyczna w szkolnictwie wyższym znajduje się w fazie intensywnej transformacji, na którą wpływają technologie cyfrowe i zmieniające się oczekiwania rynku pracy. W najbliższych latach można spodziewać się dalszej ewolucji modeli dydaktycznych, ukierunkowanej na pełniejsze wykorzystanie sztucznej inteligencji, immersyjnych środowisk symulacyjnych oraz elastycznych form potwierdzania kompetencji. Przegląd literatury wskazuje, że rozwój edukacji IT będzie opierał się na czterech strategicznych trendach (OECD, 2023b; Radianti i in., 2020).

- Personalizacja uczenia z wykorzystaniem AI.

Rozwój systemów edtech opartych na sztucznej inteligencji umożliwia tworzenie spersonalizowanych ścieżek edukacyjnych, dostosowanych

do stylu uczenia się, poziomu kompetencji oraz tempa pracy studenta. AI analizuje aktywność użytkownika, diagnozuje braki kompetencyjne i rekomenduje treści oraz zadania dopasowane do indywidualnych potrzeb. Najważniejsze kierunki rozwoju obejmują: inteligentnych tutorów wspierających naukę programowania (Denny i in., 2023), systemy adaptive learning, które automatycznie dostosowują poziom trudności zadań, asystentów AI analizujących kod, popełniane błędy i proponujących korekty, automatyczną analizę stylu pracy grup projektowych, w tym wkładu poszczególnych członków zespołu. Badania wykazują, że spersonalizowana edukacja oparta na danych zwiększa efektywność uczenia nawet o 20-35% w porównaniu z jednostandardowymi modelami dydaktycznymi (Xie i in., 2019).

- Symulacje VR i rozszerzona rzeczywistość w dydaktyce technicznej. Wirtualna rzeczywistość (VR) oraz rzeczywistość rozszerzona (AR) są coraz częściej wykorzystywane w szkolnictwie wyższym, zwłaszcza w dziedzinach technicznych. Ich największym atutem jest możliwość odwzorowania kosztownych, niebezpiecznych lub trudno dostępnych środowisk technicznych (Radianti i in., 2020). Przykłady zastosowań VR/AR w edukacji informatycznej: symulacje środowisk sieciowych i centrów danych, wizualizacja struktur algorytmicznych i złożonych procesów obliczeniowych, trening cyberbezpieczeństwa w warunkach zbliżonych do rzeczywistych (np. sandbox attack simulations), projektowanie i testowanie aplikacji AR/VR w laboratoriach wirtualnych. Symulacje immersyjne zwiększają zaangażowanie studentów oraz efektywność przyswajania wiedzy proceduralnej. Zgodnie z metaanalizami (Radianti i in., 2020), VR podnosi skuteczność nauki o 25-30% w porównaniu z metodami tradycyjnymi w obszarach wymagających uczenia manualnego lub proceduralnego.
- Mikrocertyfikaty i elastyczne ścieżki potwierdzania kompetencji. W odpowiedzi na dynamiczne zmiany technologiczne uczelnie coraz częściej sięgają po mikrokredencjały (microcredentials), czyli krótkie certyfikaty potwierdzające opanowanie konkretnej umiejętności. Najważniejsze korzyści: większa elastyczność dla studentów i pracowników, możliwość zdobywania kompetencji dokładnie wtedy, gdy są

potrzebne (just-in-time learning), współpraca uczelni z firmami technologicznymi (np. AWS, Google, Microsoft), szybkie uzupełnianie kwalifikacji wymaganych w projektach IT. Według badania European Commission (2022), rynek mikrokredencjałów podwoił się w latach 2020-2022, a ponad 70% uczelni wyższych planuje ich dalsze wdrażanie. Typowe mikrocertyfikaty w informatyce: certyfikaty chmurowe (AWS Cloud Practitioner, Azure Fundamentals), DevOps Foundations, Introduction to Cybersecurity, Data Analytics Fundamentals. Mikrocertyfikaty zwiększają szanse absolwentów na znalezienie pracy, ponieważ potwierdzają aktualne umiejętności bardziej precyzyjnie niż tradycyjne dyplomy.

- Automatyzacja laboratoriów i integracja z DevOps.

Kierunkiem o rosnącym znaczeniu jest pełna automatyzacja laboratoriów informatycznych, wykorzystująca technologie takie, jak: Infrastructure as Code (IaC), CI/CD, konteneryzacja (Docker), orkiestracja usług (Kubernetes). Automatyczne laboratoria umożliwiają studentom realizację zadań w środowiskach identycznych z tymi stosowanymi w przemyśle IT (Zhang i in., 2021). Dzięki temu proces dydaktyczny staje się bliższy rzeczywistym praktykom inżynierskim. Przykłady rozwiązań: GitHub Classroom z integracją CI/CD, wirtualne klastry Kubernetes dla zajęć laboratoryjnych, automatyczne testowanie aplikacji w pipeline'ach, generowanie środowisk na żądanie (on-demand labs). Automatyzacja laboratoriów: skraca czas konfiguracji środowiska o 70-90%, umożliwia jednoczesne prowadzenie zajęć dla dużych grup i minimalizuje błędy związane z różnicami sprzętowymi studentów. Integracja edukacji z praktykami DevOps pozwala studentom rozwijać kluczowe kompetencje wymagane przez współczesnych pracodawców, takie jak automatyzacja, współpraca zespołowa oraz szybkie wdrażanie zmian w oprogramowaniu.

Podsumowanie

Nowoczesne metody edukacji informatycznej w szkolnictwie wyższym stanowią odpowiedź na dynamiczne zmiany zachodzące w sektorze

technologii cyfrowych. Współczesna dydaktyka odchodzi od modelu transmisyjnego, opartego na biernym przyswajaniu wiedzy, na rzecz podejść aktywizujących, które angażują studentów w proces samodzielного badania, eksperymentowania i rozwiązywania problemów (Biggs, Tang, 2011). Szczególną rolę odgrywają metody projektowe (project-based learning), umożliwiające zdobywanie kompetencji praktycznych oraz rozwijanie umiejętności współpracy w zespołach, co jest zgodne z oczekiwaniami rynku pracy IT (Thomas, 2020).

Rozwój technologii chmurowych oraz narzędzi do wirtualizacji środowisk laboratoryjnych pozwala na prowadzenie zajęć w sposób skalowalny, dostępny i zautomatyzowany. Dzięki temu uczelnie mogą tworzyć nowoczesną, elastyczną infrastrukturę dydaktyczną, umożliwiając studentom pracę w warunkach zbliżonych do realnych środowisk produkcyjnych. Jednocześnie coraz większe znaczenie zyskują narzędzia sztucznej inteligencji, wspierające proces nauczania (np. automatyczna analiza kodu, generowanie feedbacku) i uczenia się, poprzez personalizację ścieżek edukacyjnych oraz inteligentne systemy tutoringowe (Marinescu, 2017).

Ważnym wyzwaniem pozostaje przygotowanie kadry akademickiej do transformacji dydaktycznej – zwłaszcza w zakresie kompetencji cyfrowych, pracy z narzędziami AI oraz projektowania kursów praktycznych wymagających interdyscyplinarnego podejścia (Redecker, 2020). Równie istotne jest zapewnienie równowagi między automatyzacją a rozwijaniem kompetencji manualnych i poznawczych. Nadmierne poleganie na narzędziach automatyzujących analizę kodu czy generowanie rozwiązań może ograniczać rozwój samodzielności oraz umiejętności krytycznego myślenia, kluczowych dla profesjonalistów IT (Shute, Wang, 2022).

Mimo tych wyzwań wdrażanie nowoczesnych metod dydaktycznych przynosi wymierne korzyści. Badania wskazują na zwiększoną motywację studentów, wyższą efektywność przyswajania materiału oraz lepsze przygotowanie absolwentów do pracy w branży technologicznej (Fischer et al., 2020). Integracja metod projektowych, środowisk chmurowych, narzędzi AI oraz pracy zespołowej pozwala uczelniom budować programy studiów bardziej dopasowane do realiów współczesnego rynku, a studentom – rozwijać kompetencje praktyczne i społeczne niezbędne w zawodach przyszłości.

Literatura

1. Behera, S., Dash, S. (2016). Project-based learning and its impact on engineering education. *International Journal of Engineering Research*.
2. Biggs, J., Tang, C. (2011). *Teaching for Quality Learning at University*. McGraw-Hill.
3. Bishop, J. L., Verleger, M. A. (2013). *The flipped classroom: A survey of the research*. ASEE National Conference.
4. Bond, M., Buntins, K., Bedenlier, S., Zawacki-Richter, O., Kerres, M. (2021). *Digital transformation in higher education: Challenges and opportunities*. Springer.
5. Caballé, S., Conesa, J. (2018). *Formative assessment, learning data analytics and gamification in ICT education*. Springer.
6. Chen, X., Xie, H., Zou, D., Hwang, G. J. (2020). Application and theory of artificial intelligence in education. *Educational Technology & Society*, 23(3), 14-28. <https://doi.org/10.1016/j.caeai.2020.100002>
7. Cotton, D., Cotton, P., Shipway, J. (2023). Chatting and cheating: Ensuring academic integrity in the era of ChatGPT. *Innovations in Education and Teaching International*, 1-12. <https://doi.org/10.1080/14703297.2023.2190148>
8. Deterding, S., Dixon, D., Khaled, R., Nacke, L. (2011). *From game design elements to gamefulness: Defining gamification*. MindTrek Conference.
9. Dziuban, C., Graham, C. R., Moskal, P., Norberg, A., Sicilia, N. (2018). Blended learning: The new normal and emerging technologies. *International Journal of Educational Technology in Higher Education*, 15(3), 1-16. <https://doi.org/10.1186/s41239-017-0087-5>
10. EDUCAUSE. (2022). *Horizon Report: Information Security Edition*. EDUCAUSE Publications.
11. European Commission. (2022). *Microcredentials for lifelong learning and employability*.
12. Ferrari, A. (2013). *DIGCOMP: A framework for developing and understanding digital competence in Europe*. Publications Office of the European Union.
13. Fischer, C., Pardos, Z. A., Baker, R. S. (2020). The Impacts of Adaptive Learning on Student Performance. *Computers & Education*, 157, 103987.
14. Floridi, L., Cowls, J. (2019). *A unified framework of five principles for AI in society*. Harvard Data Science Review. <https://doi.org/10.1162/99608f92.8cd550d1>
15. Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., Wenderoth, M. P. (2014). *Active learning increases student performance in*

- science, engineering, and mathematics. PNAS, 111(23), 8410-8415. <https://doi.org/10.1073/pnas.1319030111>*
16. Goel, A., Polepeddi, L. (2016). *Jill Watson: A Virtual Teaching Assistant for the Classroom*. Georgia Institute of Technology Research Reports.
17. Graham, C. (2019). *Current trends in blended learning in higher education*. ECAR.
18. Hmelo-Silver, C. E. (2004). Problem-based learning: What and how do students learn? *Educational Psychology Review, 16(3)*, 235-266. <https://doi.org/10.1023/B:EDPR.0000034022.16470.f3>
19. Hrastinski, S. (2019). What do we mean by blended learning? *TechTrends, 63(5)*, 564-569.
20. Keuning, H., Jeurig, J., Heeren, B. (2018). A systematic literature review of automated feedback for programming exercises. *ACM Transactions on Computing Education, 19(1)*, 1-43. <https://doi.org/10.1145/3231711>
21. Kim, G., Humble, J., Debois, P., Willis, J. (2016). *The DevOps handbook*. IT Revolution Press.
22. Kirkwood, A., Price, L. (2014). Technology-enhanced learning and teaching in higher education: Critical review. *Learning, Media and Technology, 39(1)*, 6-36. <https://doi.org/10.1080/17439884.2013.770404>
23. Kop, R., Hill, A. (2008). Connectivism: Learning theory of the future or vestige of the past? *The International Review of Research in Open and Distributed Learning, 9(3)*, 1-13. <https://doi.org/10.19173/irrodl.v9i3.523>
24. Laurillard, D. (2012). *Teaching as a design science: Building pedagogical patterns for learning and technology*. Routledge.
25. Marín, V. (2021). Web-based coding environments in higher education: Effects on student engagement and performance. *Computers & Education*.
26. Marinescu, D. (2017). *Cloud Computing: Theory and Practice*. Morgan Kaufmann.
27. Nakata, K., Ravu, M., Ali, S. (2020). Communication challenges in software development teams. *Journal of Systems and Software, 170*, 110729.
28. OECD. (2023a). *Education at a Glance*. OECD Publishing.
29. OECD. (2023b). *Skills Outlook 2023: Learning in the digital age*. OECD Publishing.
30. Pachler, N., Bachmair, B., Cook, J. (2010). *Mobile learning: Structures, agency, practices*. Springer.
31. Papastergiou, M. (2009). Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student

- motivation. *Computers & Education*, 52(1), 1-12. <https://doi.org/10.1016/j.compedu.2008.06.004>
32. Piaget, J. (1970). *Science of education and the psychology of the child*. Orion Press.
33. Prince, M. J., Felder, R. M. (2006). Inductive teaching and learning methods: Definitions, comparisons, and research bases. *Journal of Engineering Education*, 95(2), 123-138. <https://doi.org/10.1002/j.2168-9830.2006.tb00884.x>
34. Radianti, J., Majchrzak, T. A., Fromm, J., Wohlgenannt, I. (2020). A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. *Computers & Education*, 147, 103778. <https://doi.org/10.1016/j.compedu.2019.103778>
35. Redecker, C. (2017). *European framework for the digital competence of educators: DigCompEdu*. Publications Office of the European Union.
36. Redecker, C. (2020). Digital Competence of Educators. *European Framework for the Digital Competence of Educators (DigCompEdu)*. Publications Office of the European Union.
37. Savery, J. R. (2015). *Overview of problem-based learning: Definitions and distinctions*. Interdisciplinary Journal of Problem-Based Learning.
38. Selwyn, N. (2016). *Education and Technology: Key Issues and Debates*. Bloomsbury.
39. Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research*, 78(1), 153-189. <https://doi.org/10.3102/0034654307313795>
40. Shute, V., Wang, L. (2022). The role of formative feedback in developing computational thinking. *Computers & Education*, 181, 104447.
41. Siemens, G. (2005). Connectivism: A learning theory for the digital age. *International Journal of Instructional Technology and Distance Learning*, 2(1), 3-10.
42. Thomas, J. (2020). Project-based learning in computer science education: A systematic review. *ACM Transactions on Computing Education*, 20(4), 1-25.
43. Thomas, J. W. (2000). *A review of research on project-based learning*. Autodesk Foundation.
44. Vaithilingam, P., Zhang, Y., Ross, B. (2022). *How developers use AI-assisted code completion tools*. ACM CHI Conference.
45. Vuorikari, R., Punie, Y., Carretero Gomez, S., Van den Brande, G. (2022). *DigComp 2.3: The Digital Competence Framework for Citizens*. European Commission.
46. Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.

47. Wada, I. U., Izibili, G. O., Babayemi, T., Abdulkareem, A., Macaulay, O. M., Emadoye, A. (2025). AI-driven cybersecurity in higher education: A systematic review and model evaluation for enhanced threat detection and incident response. *World Journal of Advanced Research and Reviews*, 25(3), 2233-2245. <https://doi.org/10.30574/wjarr.2025.25.3.0989>
48. Watson, W. R., Watson, S. L. (2012). *The role of learning management systems in personalized learning*. TechTrends.
49. Williams, L., Kessler, R. (2003). *Pair programming illuminated*. Addison-Wesley.
50. Winkler, R., Söllner, M. (2018). *Unleashing the Potential of Chatbots in Education: A State-of-the-Art Analysis*. Academy of Management Annual Meeting Proceedings.
51. World Economic Forum. (2023). *Future of Jobs Report 2023*. WEF.
52. Xie, H., Chu, H. C., Hwang, G. J., Wang, C. C. (2019). Trends and development in artificial intelligence and adaptive learning research. *Interactive Learning Environments*, 27(5-6), 956-971. <https://doi.org/10.1080/10494820.2018.1495652>
53. Xu, L., Huang, D., Tsai, W. T. (2014). Cloud-based virtual laboratory for network security education. *IEEE Transactions on Education*, 57(3), 145-150. <https://doi.org/10.1109/TE.2013.2282285>
54. Zhong, R., Tian, Z., Zhuang, Y. (2020). Cloud-based virtual laboratories in computing education: A survey. *IEEE Access*, 8, 217202-217214. <https://doi.org/10.1109/ACCESS.2020.3037067>

Netografia

1. Denny, P., MacNeil S., Savelka J., Porter L., Luxton-Reilly A. (2023). *Desirable Characteristics for AI Teaching Assistants in Programming Education*. Pobrane z: <https://arxiv.org/html/2405.14178v1> (data dostępu: 01.09.2025).
2. Downes, S. (2007). *What connectivism is*. Half an Hour. Pobrane z: <https://halfanhour.blogspot.com/2007/02/what-connectivism-is.html> (data dostępu: 01.09.2025).