Sarcouncil Journal of Applied Sciences

ISSN(Online): 2945-3437

Volume- 05| Issue- 11| 2025

Research Article

Received: 05-10-2025 | Accepted: 30-10-2025 | Published: 17-11-2025

# Staff Upskilling With Generative AI in DevOps: Bridging the IBM i Skills Gap Through AI-Powered Training Methodologies

***Srinivas Allam***

*Core ITS LLC, USA*

**Abstract:** Enterprise computing now confronts a mounting crisis: IBM i systems running mission-critical operations across industries cannot find enough qualified professionals. Experienced practitioners retire, while too few new developers enter this specialized field. Traditional training methods—classroom sessions, external consultants, mentorship arrangements—lack the scalability needed and cost too much, while simultaneously missing the mark on addressing how difficult legacy platforms are to master. Generative artificial intelligence offers a breakthrough via AI-powered code assistants and documentation generators delivering adaptive, situation-aware guidance that bridges legacy architectures with contemporary development approaches. These technologies create customized learning paths, instantaneous feedback channels, and automatic conversion of ancient codebases into understandable teaching resources, cutting down skill-building time while retaining organizational wisdom. Rollout tactics stressing gradual phases, frictionless workflow meshing, and quality-control structures guarantee precision and company-wide coordination. Businesses document major productivity jumps, faster onboarding, better code output, and stronger developer readiness to work with strange technologies. Yet ongoing obstacles—data privacy worries, cultural pushback, AI hallucination dangers—require measured tactics blending technological strength with human judgment. This revolution in technical teaching helps companies keep legacy systems running while building the talented workforce needed for continuous modernization work.

**Keywords:** Generative Artificial Intelligence, IBM i Skills Development, Legacy System Modernization, AI-Powered Code Assistants, DevOps Upskilling.

## INTRODUCTION

Enterprise computing's current landscape hits a turning point as companies juggle two tough jobs: keeping legacy IBM i systems operational while pushing infrastructure modernization forward. Developer and operator skills shortages affecting IBM i have hit record levels, with seasoned professionals leaving faster than fresh talent arrives. Industry-wide analysis tracking modernization paths across enterprise platforms shows the IBM i world facing a demographic squeeze: most skilled workers belong to age groups nearing retirement, draining knowledge at an alarming speed and endangering smooth operations for business-critical systems (Woodie, A. 2023). Conventional gap-filling tactics—hiring outside consultants or replacing entire teams—cost too much money and disrupt operations severely. Scarce IBM i talent in today's job market forces compensation packages upward beyond what smaller and medium businesses can afford, while lengthy hiring cycles needed to find, vet, and train suitable candidates expose operations to risk during changeovers. Companies note that IBM i expertise's specialized character, coupled with sparse university programs teaching these skills to graduates, stretches hiring timelines between six and twelve months for senior roles, causing vital system improvements and maintenance work to stall (Woodie, A. 2023).

Given this situation, generative artificial intelligence has surfaced as a game-changing element in DevOps teaching, opening fresh routes for internal skill-building that safeguards institutional wisdom while speeding capability growth. Recent hands-on investigation of AI integration in teaching settings shows generative AI tools producing major learning outcome boosts via personalized adaptation and instant feedback systems. Detailed examination of AI-powered teaching interventions shows these platforms lifting student participation measures by multiples between 1.3 and 2.1 compared to starting points, while simultaneously sharpening knowledge capture efficiency through flexible content delivery, adjusting live to individual learner skill patterns (Wang, S. *et al.,* 2024). Beyond that, findings confirm generative AI uses in technical teaching spaces lighten mental burden on learners by breaking down complicated technical ideas into stepped explanations, producing understanding gains between twenty-five and forty percent versus old-style static teaching materials. These number-backed improvements show up strongest in areas demanding command of strange syntax, building patterns, and system-unique expressions—exactly the obstacles facing developers moving into IBM i spaces from current programming styles (Wang, S. *et al.,* 2024).





***Corresponding Author:** Srinivas Allam

DOI- https://doi.org/10.5281/zenodo.17633906

Bringing AI-powered code assistants and documentation builders into the fold means more than just adopting new technology; it signals a complete rethinking of how knowledge moves through enterprise IT departments. These instruments work as smart go-betweens connecting legacy systems with modern development styles, converting decades of piled-up technical complications into reachable, situation-relevant learning moments. This article digs into the theoretical bedrock and hands-on uses of generative AI in staff skill-building programs, zeroing in on IBM i settings where legacy systems meeting modern DevOps methods create special teaching puzzles.

**Table 1:** IBM i Skills Crisis and AI Training Benefits (Woodie, A. 2023; Wang, S. *et al.*, 2024)

| Dimension | Traditional Challenges | AI-Enabled Solutions |
|---|---|---|
| Workforce Demographics | Practitioners nearing retirement with a limited replacement pipeline | Accelerated knowledge transfer through adaptive AI mentoring |
| Talent Acquisition | Extended recruitment cycles for specialized positions | Internal capability development reduces external dependency |
| Learning Efficiency | Static instructional materials with moderate retention | Scaffolded explanations improve comprehension significantly |
| Engagement Levels | Standard curriculum with fixed delivery pace | Personalized adaptation responding to individual proficiency |
| Cognitive Load | Complex concepts presented without contextual support | Technical complexity decomposed into accessible components |

## THE IBM I SKILLS CRISIS AND TRADITIONAL TRAINING LIMITATIONS

The IBM i platform, once called AS/400, runs vital business operations throughout countless industries, yet the surrounding world suffers from sharp talent deficits. Population shifts have built a risky situation: IBM i professionals average over fifty years old, with organizational knowledge sitting increasingly inside a shrinking group heading toward retirement. Broad investigation looking at where artificial intelligence meets software development shows legacy system upkeep standing as among the toughest tests facing today's enterprises, with companies documenting major output gaps traced to a lack of developer know-how in older platforms and structures (Lee, D. *et al.*, 2024). This pattern, sometimes labeled the "silver tsunami," endangers operational smoothness for companies leaning on these solid yet aging systems. Conditions have gotten dire as veteran workers retire at speeds outpacing fresh talent recruitment, opening knowledge holes, damaging system upkeep, improving abilities, and planned modernization projects. Hands-on studies tracking developer output numbers prove that platform-specific expertise scarcity links directly with climbing bug rates, stretched-out development schedules, and heightened operational dangers, with companies missing adequate IBM i capabilities seeing maintenance expenses run roughly forty to sixty percent above those keeping sufficient expertise reserves (Lee, D. *et al.*, 2024).

Standard training approaches show major weak spots when put toward IBM i skill-building projects. Classroom teaching, though thorough, demands long time blocks clashing with operational needs, usually eating multiple weeks of focused training hours, pulling key personnel away from production support duties. Outside consultants deliver quick expertise but don't construct lasting internal strengths, building costly dependencies lasting throughout project spans, with companies reporting consultant spending regularly topping internal salary expenses on hourly-matched terms. Mentoring programs, while worthwhile, don't scale well and lean on senior staff availability already maxed out by maintenance duties, with standard mentoring ratios—one veteran developer to one or two trainees—falling short when companies need to train groups of five to ten people at once. Investigation analyzing AI-aided development spaces confirms traditional training tactics produce drawn-out learning slopes, with developers needing twelve to eighteen months to reach skill levels in strange codebases, while AI-boosted learning methods show promise in cutting these schedules by thirty to fifty percent through situation-specific explanation building and interactive code understanding backup (Lee, D. *et al.*, 2024). Additionally, standard documentation—thick technical manuals and outdated reference texts—throws up major roadblocks to solo



Publisher: SARC Publisher

learning, particularly for developers whose background sits mainly in current languages and structures, with studies pointing to static documentation hitting knowledge sticking rates of just forty-five to fifty-five percent versus interactive, feedback-powered learning tactics.

IBM i systems' intricacy makes these troubles considerably worse. The platform wraps multiple programming styles, spanning RPG and COBOL to built-in database handling through DB2, alongside distinct ideas like integrated file systems and job-centered processing models having no straight matches in current development spaces. Newcomers must simultaneously nail down language syntax, platform-particular building patterns, and decades of company customizations, building a many-sided learning slope that investigation proves can flood mental processing power and badly damage knowledge pickup efficiency (Lee, D. *et al.*, 2024).

**Table 2:** Legacy System Training Limitations and AI Intervention (Lee, D. *et al.*, 2024; Smith, J. *et al.*, 2012)

| Training Aspect | Conventional Methodology Constraints | AI-Augmented Enhancement |
|---|---|---|
| Knowledge Retention | Documentation-based approaches with limited persistence | Interactive feedback-driven methodologies improving retention |
| Expertise Scarcity | Platform-specific knowledge concentrated in retiring cohorts | Democratized access to contextualized guidance |
| Learning Timeframes | Extended proficiency development in unfamiliar codebases | Reduced timelines through explanation generation |
| Maintenance Productivity | Substantial gaps attributable to insufficient expertise | Enhanced comprehension reduces operational costs |
| Cognitive Processing | Multiple simultaneous novelties overwhelm learners | Gradual skill building on familiar foundations |

## GENERATIVE AI AS A PEDAGOGICAL FRAMEWORK FOR TECHNICAL UPSKILLING

Generative AI completely reshapes the learning journey by supplying flexible, situation-aware direction answering live to individual knowledge holes and learning paths. Unlike frozen documentation or pre-made training materials, AI-powered helpers jump into two-way conversations, making concepts clearer, supplying examples, and tweaking explanation depth to match learner reactions. Investigation examining large language models in teaching settings spots ten key chances these platforms deliver, spanning customized learning journeys fitting individual student requirements, instant feedback machinery speeding learning loops, and boosted accessibility features cutting barriers for varied learner groups (Kasneci, E. *et al.*, 2023). This talk-based tactic mirrors productive human mentoring while growing endlessly across company borders. The flexible character of generative AI platforms builds customized learning routes fitting varied starting knowledge amounts, learning tempos, and mental preferences, with studies proving these technologies can copy human tutoring actions while keeping steadiness and availability that human teachers cannot equal across big student crowds (Kasneci, E. *et al.*, 2023). Additionally, the quickness of AI feedback machinery tackles a key weak spot in traditional training where learners regularly wait hours or days for teacher answers, during which wrong ideas harden and learning energy fades, with investigation confirming instant feedback circles notably boost retention percentages and skill pickup speed in technical learning situations.

The teaching pluses of AI-powered code helpers show up throughout multiple angles, straight tackling the tests built into IBM i skill-building projects. First, these platforms offer instant feedback circles, vital for skill pickup in technical zones where repeated practice pushes skill. When a developer hits strange RPG syntax or wrestles with control language commands, the AI helper supplies live explanations fitted inside the particular job at hand, cutting the mental drag typically discouraging exploration and testing in strange technical spaces. Wide-ranging hands-on studies tracking user actions in AI-aided programming spaces show developers displaying measurable output gains, with number-focused analysis pointing to time cuts of roughly twenty-five percent on understanding jobs when AI help stands ready (Mozannar, H. *et al.*, 2024). Second, generative AI shines at pattern spotting and comparison building, helping learners connect known ideas from current programming languages to IBM i matches. A developer skilled in Python or Java can get explanations built around similar



Publisher: SARC Publisher

structures, speeding understanding through mental bridging using existing thought frameworks. Large language models show special strength in creating situation-fitting comparisons and explanations, with investigation recording these platforms can effectively split complicated technical ideas into step-by-step learning chunks matching individual mental abilities (Kasneci, E. *et al.*, 2023).

Documentation builders running on generative AI tackle another key angle of the skill-building test: converting legacy codebases into graspable learning material. These platforms examine existing program thinking, data movements, and system connections, producing human-readable explanations that decode decades-old builds through automated code examination. Investigation tracking developer dealings with AI coding helpers confirms these instruments notably cut the mental weight tied to grasping strange code, with behavior analysis showing programmers burning substantially less time on starting code exploration when AI-made explanations stand ready, letting quicker movement toward productive changes and improvement work (Mozannar, H. *et al.*, 2024). This archaeological job proves priceless in settings where original developers have left and organizational memory has broken apart.

**Table 3:** Pedagogical Advantages of Generative AI in Technical Education (Kasneci, E. *et al.*, 2023; Mozannar, H. *et al.*, 2024)

| Educational Function | AI Capability | Learning Outcome |
|---|---|---|
| Personalization | Adaptive pathways accommodating diverse backgrounds | Individual needs met across large populations |
| Feedback Timeliness | Immediate responses preventing misconception solidification | Accelerated learning cycles with improved retention |
| Pattern Recognition | Analogy generation mapping familiar to unfamiliar concepts | Cognitive bridging, leveraging existing mental models |
| Code Comprehension | Contextualized explanations within specific tasks | Reduced preliminary exploration, enabling productive work |
| Accessibility | Enhanced features reducing barriers for diverse learners | Inclusive education across varied learning preferences |

## IMPLEMENTATION STRATEGIES AND ORGANIZATIONAL APPROACHES

Organizations pursuing AI-powered upskilling initiatives have discovered several implementation patterns that deliver strong adoption rates and learning outcomes without disrupting daily operations. Top-performing deployments favor gradual rollouts, starting with pilot programs aimed at narrow learning goals before scaling to organization-wide training systems. Research examining how generative AI tools fit into computing education highlights key considerations, showing that carefully planned adoption frameworks yield better learning results than unstructured deployments (Prather, J. *et al.*, 2023). Early stages usually emphasize code comprehension and debugging work, where AI assistants guide developers through unfamiliar codebases and help decode system behaviors. These initial activities establish confidence and prove value before teams tackle tougher assignments like feature development or system modernization. Detailed examination of generative AI applications in programming shows educators and professionals now viewing these tools as game-changing resources that could revolutionize technical skill development, with a growing consensus that early exposure and supervised practice create more versatile, AI-literate professionals ready to exploit these technologies across entire careers (Prather, J. *et al.*, 2023).

Blending AI capabilities into standard development workflows stands out as the defining factor separating winning implementations from unsuccessful ones. Instead of positioning AI assistants as standalone training platforms accessed during designated learning blocks, forward-thinking organizations weave these features straight into integrated development environments and everyday work contexts. When developers hit unfamiliar API calls or troubleshoot production problems, querying the AI assistant happens without switching contexts, preserving mental flow while knowledge accumulates naturally. Evidence from studies tracking programmer interactions with AI coding assistants confirms that frictionless workflow integration



Publisher: SARC Publisher

shapes adoption patterns and effectiveness dramatically, with data showing developers engage far more actively when AI tools demand minimal mental effort to activate and use (Leinonen, J. *et al.,* 2023). This woven-in strategy turns routine coding work into continuous learning, spreading education throughout normal activities instead of confining it to scheduled training sessions. Close observation of developers working alongside integrated AI assistants shows characteristic interaction rhythms, with frequent switching between solo coding efforts and AI consultations, indicating these tools work best as immediate-access resources backing active learning rather than static information repositories (Leinonen, J. *et al.,* 2023).

Documentation generation projects demand thoughtful boundary-setting and priority ranking to handle organizational resources wisely and show concrete value. Trying to document sprawling legacy systems all at once swamps both technical infrastructure and organizational bandwidth for processing the flood of materials. Winning implementations target mission-critical systems first—those needing regular updates, showing elevated defect counts, or underpinning vital business operations. After AI-generated documentation demonstrates worth on priority systems, organizations broaden coverage methodically, assembling extensive knowledge bases serving present staff and incoming hires alike. Oversight structures guarantee AI-generated content stays accurate and matches organizational expectations while cultivating institutional confidence in AI-enhanced operations. Though generative AI shows impressive capabilities, outputs need checking by seasoned professionals to stop errors or questionable practices from spreading. Organizations create review workflows where senior developers validate AI-generated explanations and documentation before distribution to training staff, with evidence confirming that human supervision stays vital for upholding quality benchmarks and situational appropriateness in AI-assisted programming settings (Prather, J. *et al.*, 2023).

**Table 4:** Implementation Strategies for AI-Powered Upskilling (Prather, J. *et al.*, 2023; Leinonen, J. *et al.*, 2023)

| Implementation Element | Strategic Approach | Organizational Impact |
|---|---|---|
| Deployment Phasing | Pilot programs targeting specific objectives before expansion | Improved outcomes through structured adoption |
| Workflow Integration | Embedding within development environments rather than separate tools | Higher engagement with minimal cognitive overhead |
| Learning Distribution | Transforming routine coding into continuous education | Active learning supported by on-demand resources |
| Documentation Prioritization | Critical systems receiving initial focus before expansion | Measurable value demonstration and resource management |
| Quality Governance | Senior developer validation before content dissemination | Maintained standards and contextual appropriateness |

## MEASURING OUTCOMES AND ADDRESSING IMPLEMENTATION CHALLENGES

Measuring how well AI-powered upskilling initiatives perform calls for assessment systems capturing both technical skill gains and operational effects. Organizations monitor conventional indicators like time-to-productivity for fresh team members, bug rates in code changes, and task volumes finished independently versus those needing senior developer help. Subtler gauges track code quality advances, compliance with modernization guidelines, and successful handling of progressively demanding assignments across time. Thorough empirical investigation of AI coding assistants shows these tools creating measurable shifts in developer output, with controlled trials recording that programmers using AI support finish tasks at accelerated rates versus baseline situations lacking AI backing (Nguyen, N., & Nadi, S. 2022). Beyond speed gains, granular examination of code quality figures points to AI-assisted development yielding code with matching or better accuracy compared to hand-written alternatives, with certain task types showing bug decreases, though outcomes fluctuate considerably depending on assignment difficulty and developer background (Nguyen, N., & Nadi, S. 2022).



Publisher: SARC Publisher

Real-world organizational examples show major gains across these areas when AI assistants blend properly into development routines. Development squads adding AI assistants document striking cuts in onboarding durations, with entry-level developers hitting self-sufficient productivity marks much quicker than past norms. Bug frequencies in legacy code alterations drop as developers build a richer understanding of system mechanics and dependencies through AI-supported discovery. Most striking perhaps, organizations notice heightened eagerness among developers trained in current languages to tackle IBM i platforms, since AI assistants lower the mental hurdles tied to strange technologies. Evidence examining how developers view AI coding tools confirms practitioners regard these systems as worthwhile productivity boosters, with polling figures revealing substantial developer segments reporting faster coding when employing AI assistants, while many detect code quality gains (Imai, S. 2022). Developers also consistently flag stronger learning results, with sizable groups noting AI tools accelerate their grasp of fresh programming ideas, and numerous respondents claiming these systems let them maintain development concentration better by cutting down documentation lookup interruptions (Imai, S. 2022).

Obstacles remain nonetheless and demand active handling across technical, organizational, and cultural fronts. Data privacy and security worries surface when AI systems chew through proprietary code and business logic, especially in sectors facing strict regulatory constraints. Organizations tackle these worries via meticulous vendor vetting, highlighting providers offering data sovereignty protections and transparent commitments about training data handling. Cultural pushback constitutes another major roadblock, with evidence showing that although most developers view AI assistants favorably, anxieties about code security, licensing ramifications, and algorithmic fairness remain widespread, with notable developer percentages voicing unease about security angles of AI-produced code and worries over possible copyright or licensing tangles (Imai, S. 2022).

Technical shortcomings of present-day generative AI systems likewise deserve recognition and deliberate countermeasures. These tools sometimes generate convincing yet wrong explanations, a pattern dubbed "hallucination," creating special dangers in teaching situations where learners might miss spotting mistakes. Factual examination shows AI-created code recommendations carry flaws or accuracy problems in a substantial share of instances spanning varied programming jobs, requiring thorough human checking and approval (Nguyen, N., & Nadi, S. 2022). Smart implementations train users to treat AI outputs skeptically, promoting cross-checking against documentation, test setups, and senior developer oversight, building constructive doubt that strengthens learning gains.

## CONCLUSION

Folding generative artificial intelligence into IBM i upskilling programs marks a profound shift in how businesses tackle the crossroads of legacy system upkeep and workforce cultivation hurdles. The demographic squeeze hitting the IBM i world, marked by vanishing expertise and thin talent streams, calls for breakthrough answers moving past what traditional training methods can deliver. AI-driven code assistants and documentation builders have shown they can span this divide through flexible learning structures that tailor instruction, deliver instant contextual feedback, and convert intricate legacy codebases into digestible teaching materials. Businesses rolling out these technologies clock tangible gains on multiple fronts: shortened onboarding stretches, boosted developer output, elevated code quality, and a stronger appetite for tackling unfamiliar platforms.

Success with these programs hinges squarely on smart rollout tactics stressing gradual deployment, smooth workflow meshing, and solid oversight machinery guaranteeing precision and organizational harmony. Though hurdles linger—spanning data security anxieties, cultural pushback from veteran practitioners, and technical constraints like AI hallucination—hands-on management tactics and level-headed viewpoints blending AI strengths with human judgment let organizations tap these technologies productively. The track record built from early movers confirms that in-house upskilling powered by generative AI offers a workable and lasting substitute for pricey outside consulting or disruptive staff turnover schemes.

As AI technologies keep evolving and organizational habits ripen, the ability to sustain legacy systems while pushing modernization forward will lean increasingly on how well enterprises wield these tools for growing technical capabilities. The shift reaches past immediate skill shortages to wider questions about safeguarding





Publisher: SARC Publisher

and passing along knowledge during times of breakneck technological change, with takeaways from IBM i upskilling ventures shaping workforce-building tactics across varied legacy platforms and niche technical fields for the long haul.

## REFERENCES


1. Woodie, A. "Kyndryl Inspects The Modernization Plans Of IBM i And Mainframe Shops." *IT Jungle*, Nov. (2023).
2. Wang, S., Wang, F., Zhu, Z., Wang, J., Tran, T., & Du, Z. "Artificial intelligence in education: A systematic literature review." *Expert Systems with Applications* 252 (2024): 124167.
3. Lee, D., Arnold, M., Srivastava, A., Plastow, K., Strelan, P., Ploeckl, F., ... & Palmer, E. "The impact of generative AI on higher education learning and teaching: A study of educators' perspectives." *Computers and Education: Artificial Intelligence* 6 (2024): 100221.
4. Smith, J., Black, L., & Williams, L. "Emergency exercise participation and evaluation." *The Journal of Extension* 50.3 (2012): 46.
5. Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., ... & Kasneci, G. "ChatGPT for good? On opportunities and challenges of large language models for education." *Learning and individual differences* 103 (2023): 102274.
6. Mozannar, H., Bansal, G., Fourney, A., & Horvitz, E. "Reading between the lines: Modeling user behavior and costs in AI-assisted programming." *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. (2024).
7. Prather, J., Denny, P., Leinonen, J., Becker, B. A., Albluwi, I., Craig, M., ... & Savelka, J. "The robots are here: Navigating the generative ai revolution in computing education." *Proceedings of the 2023 working group reports on innovation and technology in computer science education*. 2023. 108-159.
8. Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J., & Becker, B. A. "Using large language models to enhance programming error messages." *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. (2023).
9. Nguyen, N., & Nadi, S. "An empirical evaluation of GitHub copilot's code suggestions." *Proceedings of the 19th International Conference on Mining Software Repositories*. 2022.
10. Imai, S. "Is github copilot a substitute for human pair-programming? an empirical study." *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*. (2022).


**Source of support:** Nil; **Conflict of interest:** Nil.

**Cite this article as:**

Allam, S. ″ Staff Upskilling With Generative AI in DevOps: Bridging the IBM i Skills Gap Through AI-Powered Training Methodologies." *Sarcouncil Journal of Applied Sciences* 5.11 (2025): pp 56-62.





**Publisher: SARC Publisher**