VJE Tạp chí Giáo dục (2026), 26(13), 70-77 ISSN: 2354-0753

XÂY DỰNG KHUNG NĂNG LỰC GIẢI QUYẾT VẤN ĐỀ TRONG LẬP TRÌNH CHO SINH VIÊN NGÀNH CÔNG NGHỆ THÔNG TIN TẠI VIỆT NAM: MỘT TIẾP CẬN TÍCH HỢP ĐA LÍ THUYẾT

CONSTRUCTING A PROGRAMMING PROBLEM-SOLVING COMPETENCY FRAMEWORK FOR INFORMATION TECHNOLOGY STUDENTS IN VIETNAM: A MULTI-THEORETICAL INTEGRATED APPROACH

Nguyễn Đình Thọ[1,+], **Nguyễn Phương Chi**[2], **Phạm Thọ Hoàn**[2]

[1]Trường Đại học Thủ Dầu Một;
[2]Trường Đại học Sư phạm Hà Nội
+ Tác giả liên hệ ● Email: thond@tdmu.edu.vn

Article history
Received: 09/4/2026
Accepted: 06/5/2026
Published: 05/7/2026

Keywords
Programming problem-solving competency, competency framework, computational thinking, multi-theoretical integrated approach, tertiary education

ABSTRACT
Programming problem-solving competence is recognized as a core competency for Information Technology students in the digital era. However, the lack of a systematic framework integrating international theories with the Vietnamese higher education context has hindered the identification of learning outcomes and the development of comprehensive assessment instruments. This study employs a theoretical research methodology and document analysis following a four-step procedure, adopting an integrative approach drawing on eight theoretical foundations to capture the multidimensional nature of Programming problem-solving competence. The study proposes a framework comprising five key components: (1) problem analysis and understanding; (2) algorithmic solution design; (3) program implementation; (4) testing and debugging; and (5) evaluation and optimization. The framework is operationalized into 20 criteria across four developmental levels, aligned with the Vietnamese Qualifications Framework and the cultural characteristics of Vietnamese education. These findings contribute to the theoretical advancement of IT education and provide a solid foundation for curriculum design and competency-based assessment. Future research can focus on developing detailed rubrics and validating the framework through empirical studies.

1. Mở đầu

Trong kỉ nguyên số, năng lực giải quyết vấn đề (NLGQVĐ) phức tạp là kĩ năng cốt lõi (World Economic Forum, 2023). Đặc biệt, đối với sinh viên (SV) ngành công nghệ thông tin (CNTT), giải quyết vấn đề (GQVĐ) trong lập trình là năng lực (NL) cốt lõi xuyên suốt quá trình học tập và làm việc (CC2020 Task Force, 2020). Tại Việt Nam, Thông tư số 17/2021/TT-BGDĐT yêu cầu xác định rõ chuẩn đầu ra theo hướng phát triển NL (Bộ GD-ĐT, 2021). Tuy nhiên, nhiều SV vẫn gặp khó khăn khi chuyển từ lí thuyết sang thực hành, nhất là với bài toán mới (McCracken và cộng sự, 2001; Robins và cộng sự, 2003). Nguyên nhân chính được chỉ ra là do thiếu một khung NL rõ ràng để định hướng việc dạy và học, cũng như thiếu các công cụ đo lường NL phù hợp (Kiesler, 2024; Vinueza-Morales và cộng sự, 2025).

Các nghiên cứu trước đây về NLGQVĐ trong lập trình tập trung vào ba hướng chính: (1) Vai trò của tư duy tính toán: lập trình không chỉ là viết mã mà còn hỗ trợ phát triển các khái niệm khoa học máy tính như trừu tượng hóa và phân rã vấn đề (Hofstede và cộng sự, 2010; Lye và Koh, 2014); (2) Hiệu quả của phương pháp dạy học tích cực: học tập dựa trên dự án có tác động tích cực đến tư duy tính toán (Chen và Yang, 2019; Xu và cộng sự, 2023); (3) Xây dựng các khung NL lập trình và tư duy tính toán (Brennan và Resnick, 2012; CC2020 Task Force, 2020; K-12 Computer Science Framework Steering Committee, 2016). Tuy nhiên, các khung này chủ yếu được thiết kế cho bối cảnh giáo dục phổ thông hoặc giáo dục đại học ở các nước phương Tây, chưa phù hợp với đặc thù giáo dục đại học tại Việt Nam, nơi mà SV chịu ảnh hưởng của văn hóa học tập Khổng giáo (CHC) (Hofstede và cộng sự, 2010; Nguyen và cộng sự, 2025). Cụ thể, đặc điểm "học theo mẫu" (pattern imitation) trong CHC cản trở chủ yếu ở bước Phân tích vấn đề, khi SV khó trừu tượng hóa bài toán mới nếu thiếu mẫu tham chiếu. Đồng thời, đặc điểm này cũng

ảnh hưởng đến bước Thiết kế giải pháp thuật toán, khi cần sáng tạo thuật toán không có trong mẫu đã học. Bên cạnh đó, các đặc điểm "ngại phản biện" và "giữ thể diện" (face-saving) cản trở bước Kiểm thử và Gỡ lỗi, do SV có xu hướng né tránh phát hiện và thừa nhận lỗi của bản thân. Điều này ảnh hưởng đến các bước Đánh giá và Tối ưu hóa, khi cần phê phán và cải tiến giải pháp của chính mình.

Tại Việt Nam, dù đã có các nghiên cứu về đổi mới dạy học lập trình và phát triển chương trình CNTT (Lê Đức Long và Phan Văn Huy, 2017; Nguyễn Quỳnh Anh và Nguyễn Hữu Tuấn, 2024; Nguyễn Tất Thắng và Bùi Thị Hải Yến, 2021) nhưng vẫn thiếu khung NLGQVĐ trong lập trình được xây dựng hệ thống. Các nghiên cứu chủ yếu tập trung vào phương pháp hoặc chương trình, chưa tích hợp đầy đủ lí thuyết quốc tế và đặc thù người học, dẫn đến thiếu chuẩn đầu ra rõ ràng, công cụ đánh giá toàn diện và khó theo dõi tiến bộ SV. Khoảng trống này được minh chứng qua ba lớp bằng chứng cụ thể: (1) Về thực trạng SV: McCracken và cộng sự (2001) ghi nhận điểm trung bình 22,89/110 điểm trên mẫu 216 SV từ 4 trường đại học quốc tế (tương đương 21% tổng điểm); (2) Về hạn chế của các framework hiện có (International Society for Technology in Education và Computer Science Teachers Association, 2011) không bao phủ các NL cài đặt, kiểm thử, tối ưu ở bậc đại học; khung CC2020 (CC2020 Task Force, 2020) không cung cấp thang đo mức độ phát triển; không framework nào có điều chỉnh theo bối cảnh CHC; (3) Về chương trình tại Việt Nam, Nguyễn Tất Thắng và Bùi Thị Hải Yến (2021) và Lê Đức Long và Phan Văn Huy (2017) đều chỉ ra chương trình CNTT Việt Nam thiên về cú pháp ngôn ngữ, thiếu nhấn mạnh phân tích bài toán, thiết kế giải thuật, kiểm thử và tối ưu hóa, sự thiếu vắng này trùng khớp với 4/5 thành phần NL mà khung này đề xuất.

Từ khoảng trống trên, nghiên cứu lựa chọn tiếp cận tích hợp đa lí thuyết, vì NLGQVĐ trong lập trình mang tính đa chiều, với tiếp cận cấu trúc NL theo mô hình Spencer và Spencer (1993), (Le Boterf, 1994, 2000); quy trình nhận thức theo khung Polya (1957) và Jonassen (2000) dựa trên nền tảng phân biệt vấn đề có cấu trúc rõ và vấn đề cấu trúc mờ của Simon (1973); nội dung tư duy tính toán tham chiếu nghiên cứu của Wing (2006, 2008); thang phân tầng kĩ năng SOLO (Biggs và Collis, 1982), Dreyfus và Dreyfus (1980); chuẩn nghề nghiệp dựa trên CC2020 Task Force (2020). Câu hỏi nghiên cứu đặt ra: "Khung NLGQVĐ trong lập trình cho SV CNTT cần có cấu trúc như thế nào để phản ánh đầy đủ các lí thuyết nền tảng quốc tế và có tiềm năng ứng dụng trong bối cảnh giáo dục đại học Việt Nam?".

2. Phương pháp nghiên cứu

Nghiên cứu sử dụng phương pháp nghiên cứu lí luận kết hợp phân tích tài liệu (document analysis). Đây là phương pháp phù hợp cho mục tiêu xây dựng khung NL ở giai đoạn lí thuyết, trước khi tiến hành xác nhận bằng phương pháp chuyên gia và thực nghiệm sư phạm (Bowen, 2009), cho phép tổng hợp, so sánh và tích hợp các lí thuyết nền tảng một cách có hệ thống để đề xuất một mô hình lí thuyết mới.

Các tài liệu được thu thập từ các cơ sở dữ liệu học thuật quốc tế (Scopus, Web of Science, ACM Digital Library, IEEE Xplore, ERIC) và các nguồn trong nước (Tạp chí Giáo dục, Google Scholar), sử dụng các từ khóa chính: "*programming problem-solving", "computational thinking framework", "programming competency", "SOLO taxonomy programming", "competency framework IT students"*. Tiêu chí (TC) chọn tài liệu bao gồm: (1) liên quan trực tiếp đến NLGQVĐ trong lập trình hoặc tư duy tính toán; (2) đề xuất hoặc vận dụng mô hình lí thuyết về NL; (3) được công bố trên tạp chí có bình duyệt hoặc là sách chuyên khảo được trích dẫn rộng rãi. Quy trình sàng lọc tài liệu gồm 3 bước: (1) Tìm kiếm ban đầu từ 5 cơ sở dữ liệu quốc tế và 2 nguồn trong nước: thu được 156 tài liệu; (2) Sàng lọc theo tiêu đề, tóm tắt và phạm vi nội dung: còn 73 tài liệu đáp ứng TC; (3) Đọc toàn văn và áp dụng TC loại trừ: còn 38 tài liệu được đưa vào phân tích sâu (33 tài liệu quốc tế, 5 tài liệu trong nước.

Quy trình nghiên cứu được thực hiện qua bốn bước. *Bước 1 - Tổng quan hệ thống các lí thuyết nền tảng*: Phân tích và hệ thống hóa tám lí thuyết nền tảng được tổ chức thành sáu nhóm chức năng gồm các mô hình NL của Spencer và Spencer (1993), Le Boterf (1994, 2000); lí thuyết GQVĐ của Polya (1957) và Jonassen (2000); tư duy tính toán của Wing (2006) và (Brennan và Resnick (2012); phân loại mục tiêu học tập của Biggs và Collis (1982); mô hình phát triển kĩ năng của Dreyfus và Dreyfus (1980). Bên cạnh đó, khung chương trình đào tạo quốc tế CC2020 Task Force (2020) được tham chiếu như khung chuẩn nghề nghiệp quốc tế và thang Bloom sửa đổi của (Anderson và Krathwohl, 2001) được sử dụng bổ trợ như ngôn ngữ hành vi cho thang đo mức độ, không tính là một lí thuyết nền tảng riêng. Việc lựa chọn 8 lí thuyết này dựa trên TC: mỗi lí thuyết đóng góp một chiều cạnh riêng biệt và không thay thế được trong khung NL (cấu trúc NL, quy trình GQVĐ, nội dung tư duy tính toán, thang đo mức độ, chuẩn nghề nghiệp). Các lí thuyết liên quan khác như Kolb's Experiential Learning (Kolb, 1984) và Self-Determination Theory (Ryan và Deci, 2000) không được đưa vào vì tập trung vào quá trình học tập nói chung, không cung cấp cấu trúc đặc thù cho NL lập trình hay thang phân tầng kĩ năng theo TC quan sát được. Mối quan hệ giữa các lí thuyết và khung NL được cấu trúc theo hai trục. Trục dọc gồm 5 NL theo tiến trình Polya (1957), mở rộng bởi Jonassen (2000)

trên phân loại bài toán của Simon (1973); cấu trúc NL dựa trên mô hình tảng băng của (Spencer và Spencer, 1993; Le Boterf, (1994, 2000)), tư duy tính toán (Wing, 2006; Brennan và Resnick, 2012) và chuẩn ACM/IEEE CC2020. Trục ngang gồm 4 mức: SOLO mô tả độ phức tạp, Dreyfus giải thích chuyển bậc, Bloom cung cấp động từ vận hành hóa. *Bước 2 - Phân tích đặc điểm NLGQVĐ trong lập trình*: Xác định các thành phần, TC và chỉ số đặc trưng của NLGQVĐ trong bối cảnh lập trình dựa trên các nghiên cứu thực nghiệm (Loksa và cộng sự, 2016; McCracken và cộng sự, 2001; Robins và cộng sự, 2003). *Bước 3 - Tích hợp và phát triển khung NL:* Tích hợp các lí thuyết để đề xuất khung NL với 5 thành phần, 20 TC, 4 mức độ phát triển. Cơ chế tích hợp được thực hiện qua hai công cụ: (1) Mã hóa nội dung (content coding) theo 5 chiều phân tích: định nghĩa NL; thành phần/TC; thang mức độ; bối cảnh áp dụng; khoảng trống và hạn chế, với mỗi trong tám lí thuyết được mã hóa theo 5 chiều này trước khi tổng hợp; (2) Lập bản đồ khái niệm tích hợp (concept mapping): ánh xạ tường minh từng lí thuyết sang chiều cạnh đóng góp riêng biệt và thành phần NL cụ thể được định hình. Ví dụ: Wing (2006) định hình TC1.2 (phân rã vấn đề) và TC1.4 (trừu tượng hóa) trong NL1; nội dung TC1.3 (nhận diện mẫu) được tham chiếu theo (International Society for Technology in Education và Computer Science Teachers Association, 2011) vốn phát triển từ nền tảng của Wing; Polya (1957) định hình trục dọc (chuỗi 5 thành phần NL theo tiến trình GQVĐ); SOLO, Dreyfus và thang Bloom (sửa đổi) định hình trục ngang (thang 4 mức phát triển). Nguyên tắc chọn lí thuyết là: mỗi lí thuyết được đưa vào chỉ khi cung cấp một chiều cạnh không thể thay thế bằng lí thuyết khác trong khung. *Bước 4 - Điều chỉnh phù hợp bối cảnh Việt Nam*: Rà soát khung NL theo: (1) Chuẩn đầu ra chương trình đào tạo ngành CNTT theo Thông tư số 17/2021/TT-BGDĐT (Bộ GD-ĐT, 2021); (2) Khung trình độ quốc gia Việt Nam -VQF (Thủ tướng Chính phủ, 2016); (3) Đặc điểm chương trình đào tạo về lập trình tại một số trường đại học Việt Nam; (4) Tham chiếu các nghiên cứu về đặc điểm văn hóa học tập Khổng giáo (Hofstede và cộng sự, 2010; Nguyen và cộng sự, 2025). Kĩ thuật so sánh văn bản hệ thống được dùng để đối chiếu TC khung với Thông tư số 17/2021/TT-BGDĐT (Bộ GD-ĐT, 2021) và VQF bậc 6. Kết quả dẫn đến hai điều chỉnh: (1) rút từ 5 mức SOLO còn 4 mức phù hợp với bậc đại học; (2) diễn đạt TC bằng động từ hành vi theo Bloom để đảm bảo đo lường được.

Tính tin cậy và giá trị nghiên cứu được đảm bảo qua sử dụng đa nguồn lí thuyết uy tín; quy trình phân tích tài liệu có hệ thống, TC rõ ràng; đối chiếu khung NL với chuẩn đầu ra quốc gia và chương trình thực tế. Độ tin cậy mã hóa được tăng cường bằng kiểm tra chéo giữa các tác giả và đạt đồng thuận. Quy trình mã hóa tài liệu sử dụng coding scheme 5 chiều: (1) Định nghĩa NL; (2) Thành phần/TC quan sát; (3) Thang mức độ; (4) Bối cảnh áp dụng (bậc học, lĩnh vực, văn hóa); (5) Khoảng trống, hạn chế. Hai tác giả mã hóa độc lập 50 điểm, đạt đồng thuận ban đầu 84% (42/50). Các bất đồng được thảo luận, thống nhất trước khi tổng hợp khung, phù hợp quy trình document analysis của Bowen (2009).

3. Kết quả nghiên cứu

Khung NLGQVĐ trong lập trình được đề xuất dựa trên tích hợp đa lí thuyết. Đây là kết quả của giai đoạn nghiên cứu lí luận, cần tiếp tục được kiểm chứng qua phương pháp Delphi và thực nghiệm sư phạm trong giai đoạn tiếp theo.

3.1. Phân tích bối cảnh và căn cứ xây dựng khung năng lực

Trước khi đề xuất khung NL, nghiên cứu phân tích bốn căn cứ thực tiễn nhằm đảm bảo phù hợp với bối cảnh Việt Nam. Thứ nhất, theo khung VQF, người học cần GQVĐ phức tạp với tính tự chủ, nên khung được thiết kế 4 mức: từ đầu vào (Mức 1) đến đạt chuẩn đầu ra và tiệm cận sau đại học (Mức 4), phù hợp tích hợp vào chương trình đào tạo. Thứ hai, theo Thông tư số 17/2021/TT-BGDĐT về yêu cầu chuẩn đầu ra phải đo lường được, do đó TC được diễn đạt bằng động từ hành vi và xuyên suốt các học phần trong chương trình. Thứ ba, thực tiễn đào tạo CNTT còn thiên về viết mã, thiếu nhấn mạnh phân tích, thiết kế, kiểm thử và tối ưu; khung đề xuất bổ sung 5 thành phần NL bao phủ toàn bộ tiến trình học và đáp ứng nhu cầu thị trường. Thứ tư, trong bối cảnh văn hóa học tập Khổng giáo, SV quen học theo mẫu, nên khung công nhận đây là điểm khởi đầu, đồng thời định hướng phát triển phản tư và tự chủ qua thang 4 mức. Tổng thể, khung NLGQVĐ trong lập trình đề xuất vừa đảm bảo cơ sở lí luận, vừa phản ánh đặc thù đào tạo và văn hóa học tập tại Việt Nam.

3.2. Định nghĩa năng lực giải quyết vấn đề trong lập trình

Định nghĩa NLGQVĐ trong lập trình được xây dựng theo tiếp cận tích hợp đa lí thuyết nhằm phản ánh bản chất đa chiều, do không lí thuyết đơn lẻ nào bao quát đầy đủ : (1) *Về cấu trúc nội tại*, mô hình tảng băng phân biệt lớp nổi (kiến thức, kĩ năng) và lớp chìm (động cơ, đặc điểm, tự nhận thức); vì vậy SV có thể biết cú pháp nhưng vẫn gặp khó khi thiếu kiên trì, phản tư (Le Boterf, 1994, 2000) nhấn mạnh NL là khả năng huy động, phối hợp nguồn lực phù hợp tình huống, phân biệt giữa "có" và "hành động có NL" (2) *Về quy trình nhận thức*, Polya (1957) đề xuất các bước GQVĐ, được cụ thể hóa thành phân tích, thiết kế, cài đặt, kiểm thử và tối ưu. Dựa trên phân biệt bài toán rõ/mờ

của (Simon, 1973) và Jonassen (2000) nhấn mạnh hiểu đúng bản chất vấn đề; (3) *Về tư duy tính toán*, Wing (2006) định nghĩa tư duy tính toán là hoạt động GQVĐ, thiết kế hệ thống và hiểu hành vi con người, dựa trên các khái niệm nền tảng của khoa học máy tính, với các yếu tố cốt lõi gồm trừu tượng hóa, phân rã, tái hình thức hóa và lập luận heuristic; trên nền tảng này, (International Society for Technology in Education và Computer Science Teachers Association, 2011) cùng các nguồn giáo dục phổ thông về sau phổ biến bốn trụ cột của tư duy tính toán gồm phân rã, nhận dạng mẫu, trừu tượng hóa và thiết kế thuật toán; (4) *Về phát triển NL*, SOLO (Biggs và Collis, 1982) mô tả sự tiến bộ nhận thức theo mức độ, trong khi mô hình Dreyfus và Dreyfus, (1980) nhấn mạnh vai trò của kinh nghiệm thực hành; (5) *GQVĐ là NL cốt lõi, gắn với bối cảnh thực tiễn* (CC2020 Task Force, 2020). Tổng hợp các chiều cạnh, NLGQVĐ trong lập trình được xem là NL tích hợp, mang tính quá trình, có khả năng phát triển và định hướng ứng dụng thực tiễn.

Trên cơ sở phân tích các lí thuyết nền tảng ở trên, trong nghiên cứu này, NLGQVĐ trong lập trình được hiểu là khả năng của cá nhân trong việc huy động và kết hợp một cách linh hoạt các kiến thức, kĩ năng và thái độ để phân tích và hiểu rõ yêu cầu bài toán; thiết kế giải pháp thuật toán phù hợp; cài đặt giải pháp thành chương trình máy tính; kiểm thử và gỡ lỗi chương trình; đánh giá và tối ưu hóa giải pháp nhằm giải quyết hiệu quả các vấn đề thực tiễn bằng lập trình. Khái niệm này được xây dựng kế thừa quan niệm NL của (OECD, 2018) và các nghiên cứu về quá trình GQVĐ trong học lập trình (Loksa và cộng sự, 2016; Qian và Lehman, 2017) đồng thời tham chiếu khung tư duy tính toán của (Wing, 2006), phản ánh năm đặc trưng cơ bản: (1) *Tính tích hợp*: kết hợp đồng thời kiến thức kĩ thuật, kĩ năng thực hành và thái độ học tập, phù hợp với mô hình Spencer và Spencer (1993), Le Boterf (1994, 2000); (2) *Tính quy trình*: thể hiện qua năm hành động cốt lõi theo chuỗi có thể quan sát được, kế thừa quy trình (Polya, 1957) và mở rộng theo đặc thù lập trình; (3) *Tính bối cảnh* : được thể hiện và đánh giá trong các tình huống lập trình cụ thể, theo quan điểm của (Le Boterf, 1994, 2000) và CC2020 (CC2020 Task Force, 2020) ; (4) *Tính phát triển:* có thể hình thành và tiến triển từ mức thấp đến cao qua quá trình học tập có hướng dẫn, dựa trên thang SOLO (Biggs và Collis, 1982) và mô hình (Dreyfus và Dreyfus, 1980); (5) *Tính chuyển giao*: các thành phần cốt lõi (đặc biệt NL 1 (NL1) và NL2) không gắn với ngôn ngữ lập trình cụ thể mà có thể chuyển giao sang các ngôn ngữ và lĩnh vực ứng dụng khác nhau, phù hợp với quan điểm của (Wing, 2006) về bản chất ngôn ngữ, đó là tính độc lập của tư duy tính toán. Bên cạnh đó, định nghĩa được xây dựng cho SV CNTT bậc đại học tại Việt Nam, nhấn mạnh lập trình như bối cảnh thể hiện NL, nhằm phân biệt với các tiếp cận rộng hơn về GQVĐ (OECD, 2018) hay tư duy tính toán phổ thông. Tính đặc thù này giúp hình thành TC đánh giá đo lường được, phù hợp yêu cầu của Thông tư số 17/2021/TT-BGDĐT (Bộ GD-ĐT, 2021).

3.3. Hệ thống tiêu chí đánh giá năng lực

3.3.1. Nguyên tắc xác định tiêu chí

Việc cụ thể hóa mỗi thành phần NL thành các TC đánh giá được thực hiện theo ba nguyên tắc nhất quán xuyên suốt: (1) *Tính quan sát được*: mỗi TC phải mô tả một hành vi hoặc sản phẩm có thể quan sát và đo lường trong thực tiễn dạy học lập trình, phù hợp với yêu cầu của Thông tư số 17/2021/TT-BGDĐT và quan điểm của (Spencer và Spencer, 1993) về các chỉ số NL.; (2) *Tính không trùng lặp*: mỗi TC đo lường một khía cạnh riêng biệt của thành phần NL tương ứng, không chồng chéo với các TC khác trong cùng thành phần; (3) *Tính bao phủ*: mỗi thành phần gồm bốn TC nhằm phản ánh đầy đủ các khía cạnh cốt lõi theo lí thuyết, tránh bỏ sót. Số lượng này được xác định từ phân tích nội dung, đảm bảo mỗi thành phần có bốn khía cạnh quan sát được.

3.3.2. Căn cứ lí luận cho từng nhóm tiêu chí

Nhóm TC NL1-NL5 được xây dựng trên cơ sở tích hợp nhiều lí thuyết nhằm bao quát toàn bộ tiến trình GQVĐ trong lập trình. *Nhóm NL1- Phân tích và hiểu vấn đề (TC1.1-TC1.4):* được xác định từ tư duy tính toán của Wing (2006) và lí thuyết của Jonassen (2000). Bốn TC phản ánh các thao tác cốt lõi gồm: xác định đầu vào, đầu ra, ràng buộc, phân rã vấn đề, nhận diện mẫu và trừu tượng hóa. Đồng thời, NL1 thể hiện hai NL quan trọng: biểu diễn và cấu trúc hóa vấn đề. Các nghiên cứu thực nghiệm cho thấy SV thường thất bại ở chính bốn khía cạnh này. *Nhóm NL2- Thiết kế giải pháp thuật toán (TC2.1-TC2.4***)** : dựa trên bước "lập kế hoạch" của Polya (1957), kết hợp với tư duy tính toán và chuẩn nghề nghiệp ACM/IEEE. Các TC bao gồm lựa chọn công cụ, xây dựng và biểu diễn giải pháp, xác định cấu trúc dữ liệu và so sánh, đánh giá các phương án. Điều này phản ánh tư duy thiết kế mang tính kĩ sư, không chỉ dừng ở mức học thuật. *Nhóm NL3- Cài đặt chương trình (TC3.1-TC3.4)*: thể hiện NL lập trình thực hành. Theo (Le Boterf, 1994, 2000), NL3 bao gồm cả việc thực hiện đúng kĩ thuật và vận dụng linh hoạt trong bối cảnh cụ thể. (Brennan và Resnick, 2012) xác định các thực hành chính: viết đúng cú pháp, chuyển thiết kế thành mã và tổ chức mã nguồn. Chuẩn ACM/IEEE bổ sung yêu cầu sử dụng hiệu quả hàm và module, phản ánh mức độ

chuyên nghiệp. *Nhóm NL4: Kiểm thử và gỡ lỗi (TC4.1-TC4.4)* được xem là một NL độc lập. Theo Brennan và Resnick (2012), hoạt động này bao gồm thiết kế ca kiểm thử, phát hiện và sửa lỗi. TC sử dụng công cụ gỡ lỗi dựa trên quan điểm của Spencer và Spencer về NL quan sát được. Thực tiễn cho thấy SV gặp khó khăn trong việc phân biệt các loại lỗi, do đó các TC được tách biệt rõ ràng. *Nhóm NL5- Đánh giá và tối ưu hóa (TC5.1-TC5.4)* : tương ứng với mức nhận thức cao, dựa trên bước "nhìn lại" của Polya, các TC bao gồm đánh giá kết quả, cải tiến giải pháp và phản tư. Thang SOLO và quan điểm của Le Boterf (1994, 2000) nhấn mạnh khả năng khái quát hóa, tối ưu hóa và tự điều chỉnh như những đặc trưng cốt lõi của NL chuyên môn.

3.3.3. Tổng hợp hệ thống 20 tiêu chí

Bảng 1 trình bày hệ thống thành phần NL, với 20TC hoàn chỉnh, bổ sung cột căn cứ lí thuyết chính để làm rõ nguồn gốc của từng TC, đảm bảo tính minh bạch và khả năng kiểm chứng của khung đề xuất.

Bảng 1. Hệ thống TC đánh giá NLGQVĐ trong lập trình

Thành phần NL	Mã TC	Nội dung TC
NL1: Phân tích và hiểu vấn đề	TC1.1	Xác định đầu vào, đầu ra và điều kiện ràng buộc
	TC1.2	Phân rã vấn đề thành các vấn đề con
	TC1.3	Nhận diện mẫu và khái niệm liên quan
	TC1.4	Trừu tượng hóa vấn đề
NL2: Thiết kế giải pháp thuật toán	TC2.1	Lựa chọn cấu trúc dữ liệu phù hợp
	TC2.2	Thiết kế thuật toán logic và hiệu quả
	TC2.3	Biểu diễn thuật toán bằng mã giả/sơ đồ khối
	TC2.4	So sánh và đánh giá các phương án giải quyết
NL3: Cài đặt chương trình	TC3.1	Sử dụng đúng cú pháp ngôn ngữ lập trình
	TC3.2	Dịch thuật toán thành mã nguồn chính xác
	TC3.3	Tổ chức mã nguồn rõ ràng, dễ đọc
	TC3.4	Sử dụng hiệu quả các hàm và module
NL4: Kiểm thử và gỡ lỗi	TC4.1	Thiết kế các ca kiểm thử phù hợp
	TC4.2	Phát hiện và định vị lỗi
	TC4.3	Sửa chữa lỗi hiệu quả
	TC4.4	Sử dụng công cụ hỗ trợ gỡ lỗi
NL5: Đánh giá và tối ưu hóa	TC5.1	Đánh giá tính đúng đắn của giải pháp
	TC5.2	Phân tích hiệu quả thuật toán
	TC5.3	Cải tiến và tối ưu hóa giải pháp
	TC5.4	Phản ánh và rút kinh nghiệm

3.4. Các mức độ phát triển năng lực

3.4.1. Căn cứ lí thuyết xây dựng thang đo bốn mức

Thang đo mức độ phát triển trong khung đề xuất được xây dựng từ sự tích hợp của ba lí thuyết phân tầng nhận thức và kĩ năng, mỗi lí thuyết đóng góp một chiều cạnh riêng biệt mà hai lí thuyết còn lại không thể thay thế.

(1) *SOLO Taxonomy* (Biggs và Collis, 1982) - *Chiều phức tạp nhận thức:* Cung cấp ngôn ngữ mô tả chất lượng học tập theo mức độ phức tạp nhận thức, thay vì số lượng kiến thức. Trong lập trình, tiến trình từ làm theo mẫu đến thiết kế giải pháp mới phản ánh sự chuyển từ tư duy cụ thể sang trừu tượng. Biggs và Tang (2011) khẳng định SOLO phù hợp để xây dựng rubric đánh giá NL ở bậc đại học.

(2) *Mô hình Dreyfus* (Dreyfus và Dreyfus, 1980) - *Chiều phát triển kĩ năng thực hành:* Đóng góp cốt lõi của mô hình Dreyfus là mô tả cơ chế phát triển kĩ năng theo chuyển dịch định tính: người học không chỉ "biết nhiều hơn" mà "xử lí vấn đề khác về bản chất". Điều này định hướng thiết kế rubric, trong đó mỗi mức phải phản ánh sự thay đổi cách tiếp cận, không chỉ kết quả. Cụ thể: Mức 1 trong khung tương ứng với Novice (cần quy tắc, mẫu cứng nhắc), Mức 2 với Advanced Beginner (bắt đầu đọc bối cảnh), Mức 3 với Competent (hệ thống, tự chủ), Mức 4 với Proficient/Expert (linh hoạt, phản tư, tạo ra giải pháp mới).

(3) *Thang Bloom sửa đổi* (Anderson và Krathwohl, 2001) - *Chiều động từ hành vi quan sát được:* Đóng góp cốt lõi của thang Bloom sửa đổi đối với khung đề xuất là cung cấp hệ thống động từ hành vi có thể quan sát được để mô tả từng mức độ NL, đây là một yêu cầu bắt buộc theo Thông tư số 17/2021/TT-BGDĐT. Cụ thể, các động từ ở mức thấp ("*nhận diện", "xác định", "thực hiện theo hướng dẫn*") phù hợp với Mức 1-2 trong khung, trong khi các động

từ ở mức cao ("*phân tích", "đánh giá", "tối ưu hóa", "tạo ra*") đặc trưng cho Mức 3-4. Đặc biệt, chiều siêu nhận thức, là một khả năng nhận thức về quá trình tư duy của chính mình và cũng là cơ sở lí thuyết cho TC TC5.4 (phản ánh và rút kinh nghiệm) được đặt ở Mức 3-4 trong rubric.

(4) *Lí do tích hợp ba lí thuyết và quyết định gộp thành bốn mức:* Ba lí thuyết SOLO, Dreyfus và Bloom mang tính bổ sung: SOLO mô tả độ phức tạp nhận thức quan sát được, Dreyfus phản ánh tiến trình phát triển kĩ năng, Bloom cung cấp ngôn ngữ hành vi để đánh giá. Mỗi lí thuyết riêng lẻ đều hạn chế, nên cần tích hợp để xây dựng thang đo vừa có cơ sở nhận thức, vừa vận hành được thành rubric. Thang đo được rút còn 4 mức do mức Tiền cấu trúc ít xuất hiện và khó phân biệt trong thực tế; điều chỉnh này phù hợp với Biggs và Tang (2011) và các nghiên cứu STEM (Hsu và cộng sự, 2018). Để minh họa tính vận hành được của thang đo, ví dụ TC2.2 (Thiết kế thuật toán logic và hiệu quả) ở 4 mức: Mức 1 - "Viết được thuật toán theo mẫu đã hướng dẫn cho bài toán quen thuộc có 1 vòng lặp hoặc 1 điều kiện"; Mức 2 - "Thiết kế được thuật toán cho bài toán quen thuộc có 2-3 ràng buộc, lựa chọn đúng cấu trúc điều khiển phù hợp"; Mức 3 - "Thiết kế thuật toán độc lập cho bài toán mới ở mức trung bình, giải thích được lí do lựa chọn cấu trúc"; Mức 4 - "Thiết kế và so sánh ít nhất hai thuật toán, phân tích độ phức tạp O(n) và lựa chọn có biện luận cho bài toán phức tạp chưa gặp". Rubric chi tiết theo từng TC sẽ được xây dựng và kiểm chứng trong Giai đoạn 2 của lộ trình nghiên cứu.

3.4.2. Mô tả bốn mức độ phát triển

Bảng 2 trình bày 4 mức độ phát triển NL với mô tả tổng quát, căn cứ lí thuyết và ánh xạ theo chuỗi học phần lập trình.

Bảng 2. Các mức độ phát triển NLGQVĐ trong lập trình

Mức độ	Tên mức độ	Đặc điểm chung
Mức 1	Khởi đầu	Cần hướng dẫn chi tiết; tiếp cận vấn đề theo mẫu có sẵn; nhận diện được vấn đề đơn giản nhưng chưa có phương pháp hệ thống; viết được mã theo mẫu, thường gặp nhiều lỗi cú pháp và logic. Cụ thể: SV nhận diện (identify) được cấu trúc bài toán đơn giản (ví dụ: 1 điều kiện hoặc 1 vòng lặp) khi có mẫu hướng dẫn; áp dụng (apply) cú pháp đúng trong phạm vi mẫu đã học; thực thi (execute) chương trình với dữ liệu vào đã biết trước; chưa tự phát hiện lỗi logic khi không có gợi ý.
Mức 2	Phát triển	Bắt đầu có phương pháp; phân tích được bài toán quen thuộc; thiết kế thuật toán đơn giản; viết chương trình chạy đúng với trường hợp cơ bản; cần ít hướng dẫn hơn; bắt đầu nhận ra bối cảnh bài toán. Cụ thể: SV phân tích (analyze) được bài toán quen thuộc có 2-3 ràng buộc (phức tạp hơn Mức 1); lựa chọn (select) cấu trúc điều khiển phù hợp (ví dụ: rẽ nhánh lồng nhau, vòng lặp có điều kiện phức hợp); viết chương trình chạy đúng với tập ca kiểm thử cơ bản; nhận ra và sửa được lỗi cú pháp và lỗi logic đơn giản khi được gợi ý. Điểm khác biệt cốt lõi so với Mức 1: xử lí bài toán 2-3 thành phần thay vì 1 thành phần, ít phụ thuộc vào mẫu có sẵn hơn.
Mức 3	Thành thạo	Có phương pháp hệ thống; giải quyết độc lập bài toán mức trung bình; phân tích bài toán mới, thiết kế thuật toán hợp lí; viết chương trình đúng đa số trường hợp; kiểm thử và gỡ lỗi có chủ đích; bắt đầu đánh giá hiệu quả giải pháp.
Mức 4	Nâng cao	GQVĐ phức tạp, chưa gặp; thiết kế và so sánh nhiều phương án; viết chương trình hiệu quả, mã nguồn rõ ràng; đánh giá phê phán, tối ưu hóa giải pháp; phản tư về quá trình GQVĐ; có thể chuyển giao kiến thức sang ngôn ngữ và lĩnh vực mới.

4. Kết luận và bình luận

Nghiên cứu đã đề xuất khung NLGQVĐ trong lập trình cho SV ngành CNTT dựa trên tiếp cận tích hợp đa lí thuyết. Khung NL bao gồm 5 thành phần (Phân tích và hiểu vấn đề, Thiết kế giải pháp thuật toán, Cài đặt chương trình, Kiểm thử và gỡ lỗi, Đánh giá và tối ưu hóa), 20 TC và 4 mức độ phát triển. Về ý nghĩa khoa học, khung NL được phát triển trên cơ sở tích hợp nhiều lí thuyết quốc tế đã được kiểm chứng, đồng thời được điều chỉnh phù hợp với bối cảnh giáo dục đại học Việt Nam. Khung NL này có thể làm cơ sở lí luận cho các nghiên cứu tiếp theo về dạy học và đo lường NL lập trình. Về ý nghĩa thực tiễn, khung NL là cơ sở để: xây dựng chuẩn đầu ra các môn học lập trình theo định hướng NL; thiết kế nội dung và hoạt động dạy học nhằm phát triển NL; xây dựng bộ công cụ đánh giá NL bao gồm rubric chi tiết theo từng TC và mức độ; theo dõi và hỗ trợ sự tiến bộ của SV trong quá trình học tập.

Phân tích so sánh hệ thống cho thấy ba đóng góp nổi bật của khung đề xuất. Thứ nhất, bao phủ đầy đủ tiến trình GQVĐ với 5 thành phần, bổ sung Kiểm thử - Gỡ lỗi và Đánh giá - Tối ưu hóa, vốn thường bị bỏ qua. Thứ hai, xây dựng thang đo 4 mức có thể vận hành, tích hợp SOLO, Dreyfus và Bloom, kèm mô tả cụ thể theo từng TC, khắc

phục hạn chế của các framework quốc tế vốn thiếu phân tầng. Thứ ba, điều chỉnh theo bối cảnh văn hóa học tập Khổng giáo, công nhận "học theo mẫu" là điểm khởi đầu và định hướng phát triển phản tư, tự chủ qua các mức.

Tuy nhiên, nghiên cứu vẫn một số hạn chế nhất định: Khung NL đề xuất mới ở mức lí luận, chưa được xác nhận bằng phương pháp chuyên gia như Delphi hay kiểm chứng thực nghiệm, nên tính khả thi và hiệu lực cần tiếp tục đánh giá. Việc điều chỉnh theo bối cảnh Việt Nam chủ yếu dựa trên phân tích chính sách và tham chiếu văn hóa, chưa có dữ liệu thực nghiệm từ giảng viên và SV; thang 4 mức có thể chưa đủ chi tiết để phân biệt các mức trung gian. DO đó, hướng tiếp theo gồm xây dựng, thẩm định công cụ đánh giá (rubric, bài thực hành), thiết kế dạy học theo dự án (PjBL) và thực nghiệm kiểm chứng hiệu quả khung. Từ kết quả xây dựng khung lí thuyết qua nghiên cứu này, trong thời gian tới, cần hoàn thiện công cụ đánh giá, thu thập minh chứng thực nghiệm và triển khai các nghiên cứu xác nhận giá trị của khung NL (như xác nhận nội dung qua phương pháp Delphi; xác nhận cấu trúc qua CFA hoặc phân tích Rasch trên dữ liệu thực nghiệm và kiểm định Cronbach Alpha cho từng thành phần).

Tuyên bố về vai trò của các tác giả: *Nguyễn Đình Thọ: Lên ý tưởng nghiên cứu, xây dựng khung lí thuyết, tổng hợp tài liệu, viết bản thảo. Nguyễn Phương Chi: Giám sát, chỉ đạo quá trình nghiên cứu, góp ý và sửa chữa bản thảo. Phạm Thọ Hoàn: Góp ý phương pháp nghiên cứu, rà soát và sửa chữa bản thảo.*
Tuyên bố về GenAI và Quyền tác giả: *Trong quá trình chuẩn bị bản thảo này, các tác giả đã sử dụng Claude Sonnet 4 cho mục đích hỗ trợ tìm kiếm, tổng hợp tài liệu tham khảo và chỉnh sửa, phát hiện lỗi chính tả và ngôn ngữ. Các tác giả chịu hoàn toàn trách nhiệm về nội dung khoa học, luận điểm và kết luận của bài báo.*
Tuyên bố về xung đột lợi ích: *Các tác giả tuyên bố không có xung đột lợi ích.*
Thông tin tài trợ: *Nghiên cứu này được hỗ trợ bởi Đề án 89 về Nâng cao năng lực đội ngũ giảng viên và cán bộ quản lí các cơ sở giáo dục đại học giai đoạn 2019-2030 theo Quyết định số 89/QĐ-TTg ngày 18/01/2019 của Thủ tướng Chính phủ.*

Tài liệu tham khảo

Anderson, L. W., & Krathwohl, D. R. (Eds.) (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman.

Biggs, J. B., & Collis, K. F. (1982). *Evaluating the quality of learning: The SOLO taxonomy (structure of the observed learning outcome)*. Academic Press.

Biggs, J. B., & Tang, C. (2011). *Teaching for quality learning at university* (4th ed). McGraw-Hill Education.

Bowen, G. A. (2009). Document analysis as a qualitative research method. *Qualitative Research Journal*, *9*(2), 27-40. https://doi.org/10.3316/QRJ0902027

Bộ GD-ĐT (2021). *Thông tư số 17/2021/TT-BGDĐT ngày 22/6/2021 quy định về chuẩn chương trình đào tạo; xây dựng, thẩm định và ban hành chương trình đào tạo các trình độ của giáo dục đại học*.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the Annual Meeting of the American Educational Research Association (AERA), Vancouver, Canada.

CC2020 Task Force (2020). *Computing Curricula 2020: Paradigms for Global Computing Education*. Association for Computing Machinery. https://doi.org/10.1145/3467967

Chen, C.-H., & Yang, Y.-C. (2019). Revisiting the effects of project-based learning on students' academic achievement: A meta-analysis investigating moderators. *Educational Research Review, 26, 71-81.* https://doi.org/10.1016/j.edurev.2018.11.001

Dreyfus, S. E., & Dreyfus, H. L. (1980). *A Five-Stage Model of the Mental Activities Involved in Directed Skill Acquisition* (Report No. ORC-80-2). Operations Research Center, University of California, Berkeley.

Hofstede, G. H., Hofstede, G. J., & Minkov, M. (2010). *Cultures and organizations: Software of the mind: Intercultural cooperation and its importance for survival* (3rd ed.). McGraw-Hill.

Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

International Society for Technology in Education, & Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K-12 education*. National Science Foundation.

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development*, *48*(4), 63-85. https://doi.org/10.1007/BF02300500

K-12 Computer Science Framework Steering Committee. (2016). *K-12 computer science framework*. https://k12cs.org/

Kiesler, N. (2024). Modeling Programming Competency. In *Modeling Programming Competency* (pp. 81-89). Springer. https://doi.org/10.1007/978-3-031-47148-3

Kolb, D. A. (1984). *Experimental learning: Experience as the source of learning and development*. Prentice-Hall.

Le Boterf, G. (1994). *De la compétence, essai sur un attracteur étrange*. Éditions d'Organisation.

Le Boterf, G. (2000). *Construire les compétences individuelles et collectives: Agir et réussir avec compétence*. Éditions d'Organisation.

Lê Đức Long, Phan Văn Huy (2017). Educational Programming Language và đổi mới dạy học lập trình ở trường phổ thông. *Tạp chí Khoa học, Trường Đại học Sư phạm Thành phố Hồ Chí Minh*, *14*(1), 5-15.

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016). Programming, problem solving, and self-awareness: Effects of explicit guidance. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 1449-1461. https://doi.org/10.1145/2858036.2858252

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *SIGCSE Bulletin*, *33*, 125-180. https://doi.org/10.1145/572139.572181

Nguyen, P. T., Nguyen, K. N. V., Do, H. T. T., & Nguyen, Q. T. (2025). Confucian Educational Thought and Its Relevance to Contemporary Vietnamese Education. *Philosophies*, *10*(3). https://doi.org/10.3390/philosophies10030070

Nguyễn Quỳnh Anh, Nguyễn Hữu Tuấn (2024). Phát triển chương trình học phần "Ứng dụng công nghệ thông tin trong dạy học" ngành Giáo dục mầm non, Trường Cao đẳng Sư phạm Bắc Ninh theo định hướng phát triển năng lực số. *Tạp chí Giáo dục, 24*(số đặc biệt 2), 90-96. https://tcgd.tapchigiaoduc.edu.vn/index.php/tapchi/article/view/1698

Nguyễn Tất Thắng, Bùi Thị Hải Yến (2021). Xây dựng chương trình đào tạo đại học: Nghiên cứu trường hợp ngành Sư phạm công nghệ tại Học viện Nông nghiệp Việt Nam. *Tạp chí Giáo dục*, *509*, 59-63.

OECD (2018). *Teaching for global competence in a rapidly changing world*. OECD Publishing. https://doi.org/10.1787/9789264289024-en

Polya, G. (1957). *How to solve it: A new aspect of mathematical method* (2nd ed). Princeton University Press. https://www.degruyterbrill.com/document/doi/10.1515/9781400828678/html

Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 1-24. https://doi.org/10.1145/3077618

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, *55*(1), 68-78. https://doi.org/10.1037/0003-066X.55.1.68

Simon, H. A. (1973). The structure of ill structured problems. *Artificial Intelligence*, *4*(3), 181-201. https://doi.org/10.1016/0004-3702(73)90011-8

Spencer, L. M., & Spencer, S. M. (1993). *Competence at work: Models for superior performance.* John Wiley & Sons.

Thủ tướng Chính phủ (2016). *Quyết định số 1982/QĐ-TTg ngày 18/10/2016 phê duyệt Khung trình độ quốc gia Việt Nam*.

Vinueza-Morales, M., Rodas-Silva, J., Vidal-Silva, C., Córdova-Morán, J., & Cevallos-Ayón, E. (2025). Teaching programming in higher education: A bibliometric analysis of trends, technologies, and pedagogical approaches. *Frontiers in Education*, *10*, Article 1525917. https://doi.org/10.3389/feduc.2025.1525917

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A*, *366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118

World Economic Forum (2023). *The Future of Jobs Report 2023*. World Economic Forum. https://www.weforum.org/publications/the-future-of-jobs-report-2023/

Xu, E., Wang, W., & Wang, Q. (2023). A meta-analysis of the effectiveness of programming teaching in promoting K-12 students' computational thinking. *Education and Information Technologies*, *28*(6), 6619-6644. https://doi.org/10.1007/s10639-022-11445-2