# Codility Engineering Skills Model_

## Technical Report Version 2.1


Last Update: June 2026

# Table of Contents

# Background

Codility's work is premised on the view that effective, fair, and transparent hiring and talent management is skills-based. The case for a skills-based approach has only strengthened in recent years. By 2025, close to two-thirds of employers reported using skills-based hiring practices for new entry-level roles, and a majority applied them routinely (National Association of Colleges and Employers [NACE], 2025), reflecting a broad shift away from credential-based screening toward the direct assessment of job-relevant capability. Realizing the promise of that shift depends on a prior step: defining the skills or competencies that are relevant to effective performance in a role.

Defining job-required competencies is also a foundation of defensible talent practices. Identifying job-relevant requirements is a recognized prerequisite for the valid and defensible use of selection procedures, as set out in the Uniform Guidelines on Employee Selection Procedures (Equal Employment Opportunity Commission et al., 1978) and in the Principles for the Validation and Use of Personnel Selection Procedures (Society for Industrial and Organizational Psychology [SIOP], 2018), which the American Psychological Association has adopted as policy. Establishing validity requires multiple, converging sources of evidence rather than reliance on any single method (American Educational Research Association [AERA], APA, & National Council on Measurement in Education [NCME], 2014), a standard that has shaped the development of the model described in this report. As detailed in the Intended Use and Limitations section below, the ESM is a competency model rather than a selection procedure; it provides a defensible foundation of job-relevant constructs, but it does not by itself establish the validity of any hiring decision.

Codility's Engineering Skills Model (ESM) is a conceptual framework, developed by the company's team of tenured industrial-organizational (I-O) psychologists, that describes the higher-order, role-relevant competencies required for success across technical job families. Traditional engineering taxonomies tend to fragment work into hundreds of highly granular, tool-specific entries, treating each programming language or framework as a discrete capability. This approach poses two problems. It conflates familiarity with a tool with the ability to apply it, and it dates quickly as tools churn. The ESM instead organizes engineering work around durable competencies such as solving problems, architecting systems, building reliable software, and, increasingly, working effectively with artificial intelligence. Rather than asking whether an engineer knows Python or React, the model describes how an engineer applies knowledge to real-world challenges. The model functions both as a competency framework for customers and as a roadmap that informs how the Codility content team designs, updates, and prioritizes task creation over time.

A skills model in this domain cannot be static. Engineering work is being reshaped by artificial intelligence at a pace that few prior technologies have matched. As of 2025, 84% of developers reported using or planning to use AI tools in their work, up from 76% a year earlier, and roughly half of professional developers reported using such tools daily (Stack Overflow, 2025). Employers, in turn, expect 39% of workers' core skills to change by 2030, with AI and big data the single fastest-growing skill area (World Economic Forum [WEF], 2025). A competency model that aims to remain useful must therefore be revised deliberately and on evidence as the nature of the work shifts.

This report documents Version 2.1 of the ESM. It is intended as a standalone reference. It first summarizes the multi-year development history that produced Versions 1.0 through 2.0, then documents in detail the methodology, evidence base, and structural decisions behind the 2.1 revision. Version 2.1 is the most substantial restructuring of the model since its inception. Two converging pressures motivated it: the rapid maturation of artificial intelligence into a distinct and cross-cutting domain of engineering competence, and a broader shift across the skills landscape toward frameworks that integrate technical, cognitive, and business competencies within a single architecture. To ground the revision, the Assessment Science team conducted a structured survey of the most widely adopted skills and competency models across both the technical and non-technical landscape, supplemented by primary input from internal and external subject-matter experts (SMEs) and a confirmatory feedback survey of Codility's own engineering organization.

As in prior versions, the ESM is intentionally forward-looking and extends beyond today's content coverage. While every task on the Codility platform maps to a skill represented in the ESM, not every competency in the model has a one-to-one task available. This is by design. The model is meant to describe the competencies that matter for engineering work now and in the near future, giving organizations a stable target as technology and the nature of the work continue to evolve.

# Model Development and Evolution

The ESM has been developed through a multi-year, evidence-based process grounded in established best practices for competency modeling (Campion et al., 2011; Schippmann et al., 2000). The ESM is a competency model; its intended use, scope, and limitations, including its relationship to formal job analysis and to selection, are set out in the Structure, Intended Use, and Limitations section. The following sections summarize the evolution of the model from its foundational version through the present revision. The early history (Versions 1.0 through 2.0) is presented in condensed form to provide context; readers familiar with prior technical reports may proceed to the section titled The Version 2.1 Revision.

## Foundations: ESM 1.0 through 2.0

The first version of the model (ESM 1.0) was created to provide a structured, research-backed foundation for assessing technical talent. Though intentionally broad, it served as a critical starting point that enabled organizations to align hiring and development practices with job-relevant requirements. Its development followed three parallel tracks, one for each of the foundational skill domains, described below.

To build the technical section, the Assessment Science team examined the language-agnostic skills required across job levels for six technical job families: backend developers, frontend developers, data scientists, quality-assurance (QA) engineers, mobile developers, and developer-operations (DevOps) professionals. To ensure coverage across these families, the team reviewed more than thirty software engineering skill and competency frameworks from public and private organizations, including the IEEE Computer Society's Software Engineering Competency Model (SWECOM; Fairley, 2014), the Software Engineering Body of Knowledge (SWEBOK; Bourque & Fairley, 2014), the SEI Software Assurance Competency Model, and the Inclusive Engineering Framework, alongside professional training curricula, academic programs, and published research. The consolidated job information was distilled into a streamlined taxonomy, which the team then refined with more than twenty-five internal and twenty-two external SMEs. In job-family-specific focus groups, these SMEs provided quantitative ratings of each skill's importance and whether it was required at entry, and those ratings were used to finalize the initial list of technical skills.

In parallel, the team reviewed job information and skill-related resources to identify the cross-functional, non-technical skills that enable engineers to apply technical expertise effectively. This produced an initial set of sixteen non-technical skills grouped into two families: intrapersonal skills, which support effective functioning in a professional environment, and collaboration skills, which support working effectively with others. A panel of seven Codility engineering managers representing diverse specialties reviewed each skill's name and definition and rated whether it was essential for effective performance across representative engineering roles; skills judged essential by at least two-thirds of the panel were retained.

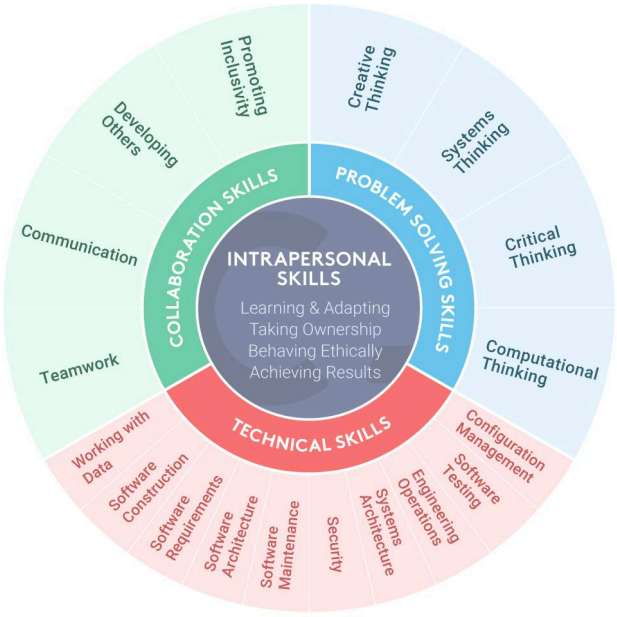
Reviewing systems-engineering competency sources, including the MITRE Systems Engineering Competency Model (Metzger & Bender, 2007) and the INCOSE Systems Engineering Competency Framework (Presland et al., 2018), the team concluded that a dedicated problem-solving category was necessary. An initial set of skills was identified, and closely overlapping constructs were consolidated, yielding the cognitive skills that anchor this category today.

To finalize ESM 1.0, the team convened a demographically diverse panel of eleven experienced internal and external SMEs, including engineering managers and individual contributors across job families together with Codility's founder, CTO, and two technical board members, and two external CTOs from midmarket and enterprise organizations. SMEs were reminded that skill and subskill names, definitions, and category labels should be broad enough to apply across roles and seniority levels, yet specific enough to characterize the profile needed for a given role, team, or project. Their written and verbal feedback was compiled into proposed changes that the team accepted, modified, or rejected by consensus, clarifying definitions, consolidating overlapping skills, reassigning subskills, and adding skills that SMEs agreed were essential. A subsequent confirmation round, together with a post-hoc comparison against the beta of IEEE's SWEBOK v4, validated the foundational model, which was presented in a wheel-shaped design emphasizing the balanced importance of technical, collaboration, and problem-solving skills.

Three subsequent versions extended this foundation, each adding a distinct layer of evidence. They also established the methodological commitments that the 2.1 revision continues: multi-source evidence, SME validation, and iterative refinement.

The first extension, released as ESM 1.1 in 2023 and 2024, addressed a key design requirement that the model capture skills essential both now and in the future. The team conducted a dedicated work analysis of AI-assisted engineering. It reviewed academic and best-practice literature on generative AI and prompt engineering, incorporated relevant work activities from O*NET, and engaged early adopters of AI coding tools through interviews, review of sample transcripts, and direct observation of SMEs working with AI systems. This work yielded twenty-three candidate work-activity statements, refined to twenty-one after comparison against O*NET baselines. A second group of six engineers with hands-on AI experience confirmed the activity-to-skill linkages, with a two-thirds agreement threshold required for each link. The analysis confirmed that many future-oriented skills were already represented in the model, while surfacing the need for a small number of new AI-related skills and reinforcing the centrality of computational thinking to AI-assisted work (Wing, 2006).

The second extension, released as ESM 1.2 in 2024, added quantitative evidence to complement the qualitative work that preceded it, since establishing validity requires both (AERA, APA, & NCME, 2014; SIOP, 2018). The team conducted a large-scale survey to capture criticality and required-upon-entry ratings for each skill across the six job families. Rather than a Likert importance scale, respondents indicated whether each skill was critical, required upon entry, or both, a more stringent standard than conventional importance ratings. The respondent pool spanned twenty-four countries, most frequently India, the United States, and Brazil. With the exception of the then-emerging AI skills, every skill was rated critical by at least a quarter of respondents in at least one job family, quantitatively confirming the relevance of the taxonomy; the newer AI skills were already considered critical for some families. These results were incorporated into the model as Version 1.2.




The third extension, released as ESM 2.0 in 2025, substantially expanded the treatment of artificial intelligence. By then it was clear that AI competence extended well beyond tool use and was no longer confined to software engineers, since the broader workforce increasingly relied on AI as well. A Q1 2025 literature review on AI readiness and AI literacy, supplemented by interviews and focus groups with internal and external technology leaders collectively representing more than 120 years of experience, established AI literacy as a multidimensional construct spanning knowledge, application, evaluation, and ethics (Long & Magerko, 2020; Ng et al., 2021). This research expanded the model with a structured set of AI-readiness skills organized into four groupings: AI Literacy, AI Evaluation, AI Application, and AI Building. SMEs additionally classified each AI skill by whether it was relevant to engineering roles specifically or to the general workforce as well, and emerging subskills such as Data Privacy were folded into existing skills. Version 2.0 housed these AI-readiness skills within the Technical domain, a placement that Version 2.1 would revisit.

# The Version 2.1 Revision

Version 2.1 was undertaken to bring the model into alignment with the current skills landscape and to resolve structural issues that had accumulated as the model grew. The revision was guided by three objectives: (a) to re-anchor the model's category architecture against the most authoritative and widely adopted skills and competency frameworks in use today; (b) to mature the treatment of artificial intelligence from a set of emerging additions into a coherent, first-class domain; and (c) to close coverage gaps and remove ambiguities identified through expert and organizational feedback. The methodology proceeded in three phases, described below.

## Phase 1: Structured Survey of the Skills Landscape

Consistent with the best-practice guidance to consider organizational context and future-oriented requirements when building a competency model (Campion et al., 2011), the Assessment Science team began by surveying the landscape of established skills and competency frameworks across both technical and non-technical domains. The objective was not to adopt any single framework wholesale, but to identify the recognized constructs against which a defensible, durable model should map. Frameworks were selected for their authority, breadth of adoption, recency, and evidentiary basis. The set spanned government job-analysis data, discipline-wide bodies of knowledge, psychometrically validated competency models, binding regulation and formal standards, current empirical labor-market signals, and the most recent observational evidence on how AI is actually used in technical work.

Several recent developments in the landscape directly shaped the revision. The IEEE Computer Society released SWEBOK Version 4.0 in October 2024, which for the first time elevated software architecture, software security, and software engineering operations to full knowledge areas (IEEE Computer Society, 2024). This independent, discipline-wide consensus validated the prominence the ESM already gave these areas within its Technical category. In the same month, the Skills Framework for the Information Age (SFIA) released Version 9, expanding to 147 leveled skills and strengthening its coverage of AI and machine-learning system development and cybersecurity (SFIA Foundation, 2024). SFIA's seven-level structure continues to inform the ESM's proficiency anchoring. Government and labor-market sources, including the U.S. Department of Labor's O*NET Content Model and Lightcast's continuously updated open skills taxonomy, provided expert-rated and empirically derived confirmation of the model's cross-cutting cognitive and business competencies and its catalog of programming languages and technologies.

Because artificial intelligence is the fastest-moving region of the landscape, the team deliberately weighted the most current sources available at the time of the revision. The NIST AI Risk Management Framework (AI RMF 1.0) organizes AI risk work into four functions, Govern, Map, Measure, and Manage, and provided the structural backbone for the model's AI governance and responsibility constructs (National Institute of Standards and Technology, 2023). The European Union's AI Act (2024) and the ISO/IEC 42001 AI management-system standard (2023) reinforced the regulatory and policy dimensions of AI competence. Anthropic's AI Fluency framework, which articulates four proficiency dimensions (Delegation, Description, Discernment, and Diligence), informed the model's treatment of human and AI interaction (Feller & Dakan, 2025). Together these sources allowed the AI category to be grounded simultaneously in proficiency, how individuals work with AI, and governance, how organizations and individuals manage its risks.

The team also drew on the newest available observational and survey evidence to confirm that the model's emphases match how engineering work is changing in real time. Anthropic's Economic Index, which analyzes millions of anonymized AI interactions, reported in early 2026 that coding remains the single most common use of its models and that roughly 49% of occupations had already seen at least a quarter of their constituent tasks performed with AI assistance (Anthropic, 2026a). A companion report found that more experienced users obtain measurably higher success rates and interact more collaboratively with AI, consistent with learning-by-doing rather than one-off tool use (Anthropic, 2026b). Practitioner survey data corroborated both the scale and the friction of adoption: 84% of developers reported using or planning to use AI tools, yet trust in AI output had fallen to 29% and nearly half cited time spent debugging AI-generated code as a primary frustration (Stack Overflow, 2025).

Recent evidence published by the major AI laboratories was especially useful in establishing the right unit of analysis for a skills model. OpenAI's GDPval evaluation assessed frontier models against the real deliverables of 44 knowledge-work occupations across nine sectors, using 1,320 tasks vetted by professionals with an average of more than fourteen years of experience, and found that model performance on these tasks roughly doubled between successive model generations and is now approaching the quality of expert human work (OpenAI, 2025). Microsoft's analysis of 200,000 anonymized Copilot conversations reached a complementary and important conclusion: generative AI most readily supports tasks involving information gathering, writing, and communication, but does not by itself fully perform any single occupation (Tomlinson et al., 2025). Read together, these laboratory studies reinforce a premise central to the ESM. AI is reshaping work at the level of tasks and competencies rather than wholesale jobs, which is precisely the level at which a competency model operates. They also align with OpenAI's framing of the present period as a workforce transition that rewards reskilling and human and AI collaboration rather than straightforward substitution (OpenAI, 2026).

At the macro level, the World Economic Forum's Future of Jobs Report 2025 identified AI and big data as the single fastest-growing skill area, followed by networks and cybersecurity and technological literacy, and projected that 39% of workers' core skills would change by 2030 (WEF, 2025). The same report identified complementary cognitive and socio-emotional competencies, including analytical and creative thinking, resilience, curiosity, and lifelong learning, as rising in importance, corroborating the continued inclusion of the ESM's Problem-Solving and Business categories alongside its technical content. The scale of the resulting reskilling need is also reflected in industry investment: Google, for example, has consolidated nearly three thousand AI and technical courses from Google Cloud, DeepMind, and its education groups into a single skilling platform aimed at the broader workforce (Google, 2025), an institutional signal that AI competence is now treated as a cross-cutting capability rather than a specialist niche.

## Phase 2: Subject-Matter Expert Input

Landscape frameworks establish recognized constructs but cannot, on their own, determine how those constructs should be named, bounded, and organized for a specific organizational context. Consistent with the principle that competency models should be expressed in the language of the organizations that use them (Campion et al., 2011), and that multiple sources of evidence are required to support validity (AERA, APA, & NCME, 2014; SIOP, 2018), the team supplemented the landscape survey with primary input from internal and external SMEs. External experts contributed an industry-wide perspective on emerging practice, while internal engineering leaders and individual contributors ensured that skill and subskill definitions reflected the realities of contemporary engineering work and remained broad enough to apply across roles and seniority levels, yet specific enough to characterize the profile needed for a given role, team, or project.

SME input was used to fill gaps that the landscape survey surfaced but did not fully resolve, particularly in the fast-moving AI domain where formal frameworks lag practice, and to adjudicate proposed changes to category structure, skill names, and definitions. As in prior versions, written and verbal feedback was compiled into proposed changes, which the team evaluated and accepted, modified, or rejected through structured consensus discussion.

## Phase 3: Organizational Confirmation Survey

In the final phase, the revised model was circulated for confirmation to Codility's internal engineering organization through a structured feedback survey. Thirty-five engineering subject-matter experts spanning multiple job families and seniority levels were invited to review the model and complete the survey. The instrument was organized around the model's three structural levels. Respondents indicated whether the five categories accurately represented the full scope of engineering competency, whether the thirty skills captured the core competencies needed in engineering roles today, and whether the subskills reflected how each skill breaks down in practice. At each level, respondents could flag specific issues using fixed categories, including that an element was missing, should be renamed, redefined, split, merged, or removed, and could provide free-text commentary explaining each flag. As noted in the Intended Use and Limitations section, this internal confirmation survey is treated as one corroborating input rather than as independent validation of the model.

The confirmation survey supported the model's overall structure. No respondent proposed removing, renaming, or redefining any subskill, indicating that the fine-grained content was well received. The substantive feedback concentrated

at the category and skill level and converged on a small number of high-leverage refinements, each of which was evaluated against the landscape evidence and SME input before adoption. Two skills addressing architecture were judged to overlap and were consolidated into a single skill, Software & Systems Architecture, retaining the full set of underlying subskills. A recurring observation that the model lacked explicit coverage of delivery constraints and Agile practice, raised independently by multiple respondents and including a concrete case in which assessment content on Agile methodologies could not be mapped to the model, led to the addition of a Project Management skill within the Business category, encompassing delivery and Agile methodologies and the management of scope, timelines, and budgets. Feedback that the technical-tooling category's label could be misread also informed its presentation. Changes supported by only a single comment, or that reflected framing rather than structural gaps, were documented and deferred rather than adopted, in keeping with the team's standard of acting on corroborated evidence.

# The Current Model (ESM 2.1)

Version 2.1 of the Engineering Skills Model comprises **30 skills** and **180 subskills**, organized into five categories. The Technical category captures the core of engineering work. The Artificial Intelligence category represents the maturing, cross-cutting domain of human and AI competence. The Problem-Solving category captures the cognitive skills that underpin innovation and adaptability. The Business category encompasses the interpersonal, collaborative, and delivery competencies that enable engineers to apply technical expertise in organizational contexts. The Technology Stack category catalogs the specific programming languages and technologies against which competence is exercised. The composition of the model is summarized in Table 1, and full definitions and subskills appear in Appendix C.

*Table 1. Composition of ESM Version 2.1 by category.*

| Category | Skills | Subskills | Focus |
|---|---|---|---|
| Technical | 9 | 49 | Core software and systems engineering competencies. |
| Artificial Intelligence | 5 | 24 | Cross-cutting human and AI competence, from fluency to engineering. |
| Problem-Solving | 4 | 0 | Cognitive skills underpinning innovation and adaptability. |
| Business | 10 | 0 | Interpersonal, collaborative, and delivery competencies. |
| Technology Stack | 2 | 107 | Catalog of programming languages and technologies. |
| Total | 30 | 180 | |

The ESM is designed to be applicable to most technical roles, and to several adjacent and non-technical roles, readily adjustable as target jobs and technologies change, and flexible across proficiency levels. The Assessment Science team continues to monitor the model, gathering research and feedback to anticipate how shifts in the field may signal changes in the skills required for effective practice. Iteration remains central to the team's approach. The 2.1 revision is itself a product of that commitment, and continued refinement will remain a primary focus of Codility's research going forward.

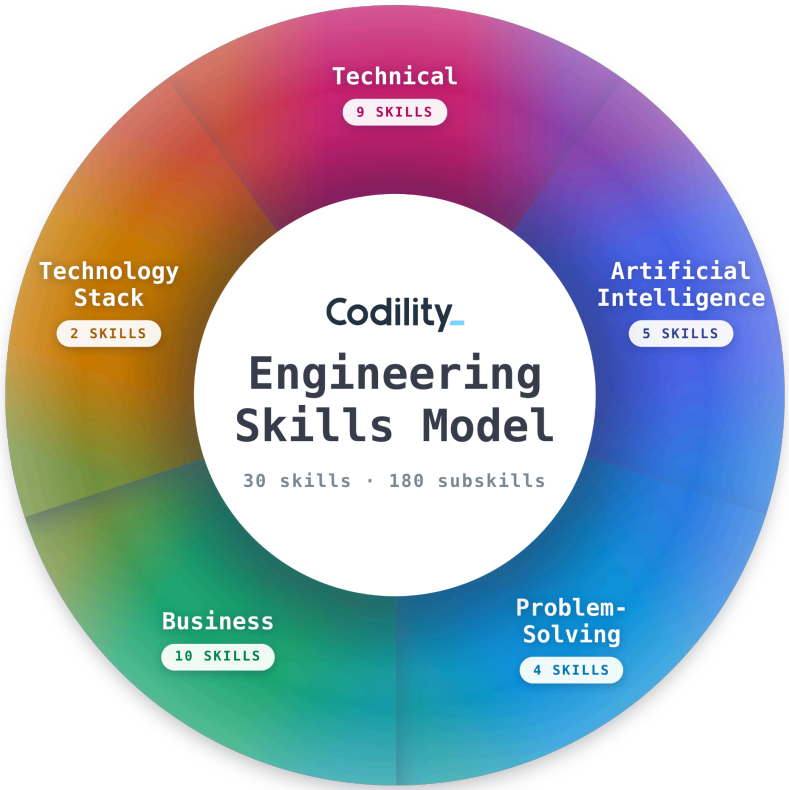
Table 2 maps each ESM category to the authoritative external frameworks and evidence that support it. This mapping supports a content-related validity argument for the model's constructs under the SIOP Principles, the AERA/APA/NCME Standards, and the EEOC Uniform Guidelines. It bears on the content representativeness of the model's constructs, not on the validity of any specific assessment built upon them.

*Table 2. Mapping of ESM 2.1 categories to authoritative external sources.*

| Source | Authority | ESM categories covered | Evidence type | Match |
|---|---|---|---|---|
| O*NET Content Model | U.S. Dept. of Labor | Problem-Solving; Business; Technical | Expert-rated job analysis | Strong |
| SWEBOK v4 (2024) | IEEE Computer Society | Technical | Expert-consensus body of knowledge | Very strong |
| SFIA v9 (2024) | SFIA Foundation | Technical; Technology Stack | Industry competency standard | Strong |
| AI Fluency (4Ds) | Anthropic (Feller & Dakan) | Artificial Intelligence (fluency, collaboration) | Practitioner framework | Strong |
| AI RMF 1.0 | NIST | Artificial Intelligence (governance, responsibility) | Standards / regulatory | Strong |
| EU AI Act; ISO/IEC 42001 | EU; ISO | Artificial Intelligence (regulation, policy) | Binding regulation / standard | Strong |

| | | | | |
|---|---|---|---|---|
| Great Eight Model | Bartram (2005) | Business; Problem-Solving | Psychometric / peer-reviewed | Strong |
| Economic Index (2026) | Anthropic | Artificial Intelligence (usage, adoption) | Empirical AI-usage data | Moderate |
| GDPval (2025) | OpenAI | Artificial Intelligence; Technical | AI capability evaluation | Moderate |
| Working with AI (2025) | Microsoft Research | Artificial Intelligence; Business | Empirical AI-usage data | Moderate |
| Future of Jobs 2025 | World Economic Forum | Problem-Solving; Business; AI | Employer survey | Moderate |
| Developer Survey 2025 | Stack Overflow | Artificial Intelligence; Technology Stack | Practitioner survey | Moderate |
| Open Skills; Hot Technologies | Lightcast; U.S. DOL | Technology Stack; Technical | Empirical labor-market demand | Strong |

*Note. Match strength reflects how closely each source's constructs correspond to the ESM's. Sources rated Strong or Very strong provide direct construct correspondence; those rated Moderate confirm relevance and demand rather than defining constructs. Empirical AI-usage and practitioner-survey sources are treated as corroborating evidence rather than as taxonomic referents.*




## Artificial Intelligence as a Cross-Cutting Domain

The most consequential structural decision in Version 2.1 was the elevation of artificial intelligence to a full, first-class category comprising five distinct skills: AI Fluency, AI Collaboration, AI Responsibility, AI Governance, and AI Engineering. In earlier versions, AI competence was represented as a set of additions distributed across the model, first as a single technical skill and later as a cluster of AI-readiness skills. The landscape survey and SME input made clear that this treatment no longer reflected the role of AI in engineering work.

Two complementary framings motivated the change. First, AI has matured into a standalone competency domain in its own right. The AI-literacy research literature consistently describes AI competence as a multidimensional construct rather than a single ability (Long & Magerko, 2020; Ng et al., 2021; Wang et al., 2023), and converging frameworks organize it around knowing and understanding AI, applying it, evaluating it, and addressing its ethical implications (Ng et al., 2021; Wang et al., 2023). Validated measurement instruments now exist for assessing these dimensions (Wang et al., 2023). The maturation of authoritative governance frameworks, including the NIST AI RMF, the EU AI Act, and ISO/IEC 42001, further establishes AI governance and responsibility as competencies that can be defined, taught, and assessed rather than treated as abstract knowledge. Organizing these into five skills allows the model to distinguish foundational fluency from collaborative practice, ethical responsibility, regulatory governance, and the specialized engineering of AI systems.

Second, AI functions as a bridge between the technical and business halves of the model. AI capability is no longer confined to specialists. Controlled field experiments with nearly five thousand developers found that access to AI coding assistants increased completed tasks by roughly 26%, with the largest gains accruing to less-experienced engineers (Cui et al., 2025). The most recent observational evidence reinforces this picture: Anthropic's Economic Index reports that coding remains the dominant use of frontier models and that roughly 49% of occupations have already seen at least a quarter of their tasks performed with AI, while coding-style capabilities increasingly diffuse to non-technical roles (Anthropic, 2026a). Evaluations from the major AI laboratories point the same way. OpenAI's GDPval study finds frontier models approaching expert-level quality on professional knowledge-work deliverables (OpenAI, 2025), while Microsoft's large-scale usage analysis finds that AI substantially supports research, writing, and communication tasks without fully performing any single occupation (Tomlinson et al., 2025). At the same time, the engineer's role is shifting from authoring every line of code toward orchestrating AI systems, validating their output, and exercising judgment about what to build. These are activities that draw as heavily on the model's Business and Problem-Solving competencies as on its Technical ones. Positioning AI as its own category, situated structurally between the Technical and Business domains, reflects this reality: AI competence is where technical depth and business judgment increasingly meet.
This framing carries a cautionary corollary that the model is designed to capture. Effective AI use is itself a learned competency, not an automatic byproduct of access to tools. Anthropic's Economic Index finds that more experienced users achieve measurably higher success rates and engage AI more collaboratively, consistent with learning-by-doing (Anthropic, 2026b), while practitioner survey data show that trust in AI output has fallen even as adoption has risen, and that debugging AI-generated code is a leading source of friction (Stack Overflow, 2025). Independent analysis likewise cautions that apparent productivity gains can be partly offset by downstream burdens such as architectural review, security auditing, code comprehension, and maintenance, leaving long-term code quality dependent on experienced human oversight (Maes, 2026). The inclusion of AI Responsibility, AI Governance, and the validation-oriented subskills within AI Collaboration reflects the team's view that responsible and discerning AI use is a distinct, measurable competency.

## Bringing Languages and Technologies into the Model

Historically, Codility treated programming languages and specific technologies as separate from skills. In earlier versions of the model, including ESM 2.0, they were not part of the framework at all; they were maintained in separate reference lists (in the prior technical report, Appendix A: Programming Languages Covered and Appendix B: Technologies Covered). The skills model described what an engineer could do, while the languages and technologies described the tools an engineer happened to use, and the two were kept deliberately distinct.

Customer conversations reshaped this view. Many organizations told us that, in practice, they treat proficiency with particular languages and technologies as core skills in their own right, inseparable from how they define and hire for a role. At the same time, the evidence and our own experience point to a transferability principle: the knowledge and disciplined application of one tool or technology carries over substantially to others in the same family. An engineer fluent in one modern web framework, relational database, or container platform can typically become productive in a comparable one far faster than someone starting from nothing, because the underlying concepts, patterns, and habits of practice transfer. Tool-specific knowledge is therefore best understood not as a collection of isolated facts but as the application of durable, transferable skill.

For these reasons, Version 2.1 brings languages and technologies into the model itself as a formal Technology Stack domain, represented by two skills, Programming Languages and Technologies, with their specific entries as subskills. It is important to be precise about what changed. Almost none of the individual languages or technologies are new content; the catalogue is substantially the same as the reference lists Codility has long maintained. What changed is structural. Items that previously lived outside the model, in standalone appendices, are now part of the model as competencies. Because the Technology Stack domain did not exist within prior versions of the model, every entry in it is, by definition, new to the model in Version 2.1, even though the underlying languages and technologies themselves are largely carried forward. This change lets the model express tool proficiency in the same competency vocabulary it uses for everything else, and it reflects how customers actually reason about engineering skill.

# Structure, Intended Use, and Limitations

## Structure and Definitions

The ESM is organized as a four-level hierarchy. A category is the broadest grouping and represents a major domain of engineering competence (for example, Technical or Artificial Intelligence). A skill is a coherent, role-relevant competency within a category, expressed as a labeled construct with a definition stating the behaviors and knowledge it encompasses. A subskill is a more specific component of a skill that describes how that skill is exercised in practice. The Technology Stack category additionally enumerates specific programming languages and technologies, which function as concrete instances against which the broader skills are applied rather than as competencies in their own right. Each ESM skill follows the standard anatomy of a competency described by Campion et al. (2011): a clear label and a behaviorally oriented definition that can be applied across roles and seniority levels.

Not all categories are decomposed into subskills. The Problem-Solving and Business categories are represented holistically at the skill level by deliberate design rather than by omission. The cognitive and interpersonal competencies they contain (for example, Critical Thinking or Communication) are understood as integrated constructs that are most validly described and assessed as wholes, and that are typically evaluated through interactive methods such as interviews rather than decomposed into discrete, separately scored components. The Technical and Artificial Intelligence categories, by contrast, lend themselves to finer decomposition because their constituent skills map more directly onto distinct, observable, and separately assessable behaviors.

## Intended Use and Scope

The ESM is a competency framework intended to support the definition of engineering competencies, the profiling of roles, learning and development, workforce and skills planning, and the design and prioritization of Codility assessment content. It provides a shared, research-grounded vocabulary for describing what engineering work requires.

The ESM is not, in itself, a selection procedure, an assessment, or a hiring decision rule, and it should not be used as one. The model defines and organizes job-relevant constructs; it does not prescribe how those constructs are measured, weighted, or combined into an employment decision. Organizations that use the ESM, or assessment content derived from it, for selection are responsible for ensuring the job-relatedness and validity of that use in their specific context and jurisdiction, including conducting a local analysis of work that links the relevant competencies to the duties of the target role, consistent with the Uniform Guidelines on Employee Selection Procedures (EEOC et al., 1978) and the SIOP Principles (2018). The ESM is informed by job-analytic sources such as O*NET, but it does not itself constitute a formal job analysis for any particular role.

## Fairness and Adverse Impact

A competency model is upstream of, and distinct from, any selection procedure, and it cannot on its own create or cure adverse impact. Adverse impact arises from the specific assessments, scoring rules, and decision thresholds that an organization applies, not from the definition of competencies. Codility makes no representation that use of the ESM, or of content derived from it, will be free of adverse impact or will guarantee compliance with any law or regulation. Where ESM-based content is used in selection, the implementing organization remains responsible for monitoring adverse impact and conducting fairness analyses as appropriate under the Uniform Guidelines (Section 4) and professional standards. The ESM is designed to support fair and transparent practice by grounding competencies in job-relevant constructs, but fairness in use is a property of the assessment system and the decisions made with it, not of the model alone.

## Nature of the Validity Evidence

The evidence presented in this report is content-oriented. The development methodology and the mapping of ESM categories to authoritative external frameworks (Table 2) support an argument that the model's constructs are representative of the domain of engineering competence. This is evidence about the content representativeness of the constructs. It is not, and should not be represented as, criterion-related or construct-validity evidence for any assessment, and it is not evidence of the validity of any selection decision. No claim of predictive validity is made for the ESM. Establishing criterion-related validity for a specific use requires a separate, local validation effort by the implementing organization.

## Proficiency and Leveling

The ESM defines competencies in a manner intended to apply across seniority levels, from early-career to principal engineers, so that the same construct can describe the skill profile required for different roles at different depths. Detailed proficiency anchoring, including the specific behavioral indicators that distinguish levels of mastery for each skill, is informed by leveled frameworks such as SFIA but is maintained separately from this report and is applied when the model is used for role profiling or assessment design. Earlier versions of the model captured entry-level requirements through criticality and required-upon-entry ratings; extending formal proficiency anchoring across all Version 2.1 skills, particularly the newer Artificial Intelligence skills, is an area of ongoing work.

## From Model to Measurement

A clear distinction should be drawn between the ESM as a model of target constructs and the assessment system that operationalizes them. Every task on the Codility platform maps to a skill represented in the ESM, but not every competency in the model is currently operationalized as a task, and the model is intentionally broader than present content coverage. Some constructs, particularly the more behavioral or dispositional ones such as Behaving Ethically, Promoting Inclusivity, and aspects of AI Responsibility, are not readily measured through automated coding tasks and are more appropriately evaluated through interviews, work samples, or structured observation. The presence of a competency in the ESM therefore indicates that it is recognized as job-relevant, not that Codility currently offers a validated measure of it. Operationalizing and validating a measure of any given competency is a separate effort governed by the same standards of evidence that informed the model itself.

## Limitations

Several limitations should be borne in mind when interpreting this report. First, the evidence supports content-oriented inferences about the representativeness of the model's constructs and does not establish criterion-related or predictive validity. Second, the Version 2.1 confirmation survey drew on a single-organization sample of internal engineering SMEs, so the absence of proposed changes at the subskill level should be read as consistent with the model rather than as strong independent endorsement. Third, the confirmation sample is a convenience sample of Codility engineers and is not necessarily representative of the broader population of engineering roles, industries, or geographies to which the model may be applied. Fourth, the model is intentionally forward-looking, so some constructs extend beyond what is empirically validated or currently measurable. Fifth, several of the Artificial Intelligence constructs are grounded partly in vendor, practitioner, and working-paper sources that have not undergone peer review and reflect a rapidly changing field; these are treated as corroborating evidence of relevance rather than as settled construct definitions, and are expected to evolve in future revisions.

## Governance and Versioning

The ESM is owned and maintained by Codility's Assessment Science team. The model is reviewed on an ongoing basis, and revisions are triggered by material shifts in the engineering or skills landscape, by accumulated expert and customer

feedback, and by the availability of new authoritative evidence. Proposed changes are evaluated and adopted, modified, or rejected through structured consensus within the team, drawing on the sources of evidence described in this report. Prior versions are archived and superseded rather than edited in place, so that the provenance of each version is preserved. Version 2.1 is the current release; continued monitoring and periodic revision will remain a primary focus of the team's research.

# References

American Educational Research Association, American Psychological Association, & National Council on Measurement in Education. (2014). Standards for educational and psychological testing. American Educational Research Association.

Anthropic. (2026a). The Anthropic Economic Index report: New building blocks for understanding AI use. Anthropic. https://www.anthropic.com/research/economic-index-primitives

Anthropic. (2026b). The Anthropic Economic Index report: Learning curves. Anthropic. https://www.anthropic.com/research/economic-index-march-2026-report

Bartram, D. (2005). The Great Eight competencies: A criterion-centric approach to validation. Journal of Applied Psychology, 90(6), 1185–1203. https://doi.org/10.1037/0021-9010.90.6.1185

Bourque, P., & Fairley, R. E. (Eds.). (2014). Guide to the software engineering body of knowledge (SWEBOK®): Version 3.0. IEEE Computer Society Press.

Campion, M. A., Fink, A. A., Ruggeberg, B. J., Carr, L., Phillips, G. M., & Odman, R. B. (2011). Doing competencies well: Best practices in competency modeling. Personnel Psychology, 64(1), 225–262. https://doi.org/10.1111/j.1744-6570.2010.01207.x

Cui, Z. K., Demirer, M., Jaffe, S., Musolff, L., Peng, S., & Salz, T. (2025). The effects of generative AI on high-skilled work: Evidence from three field experiments with software developers. Management Science. Advance online publication. https://doi.org/10.1287/mnsc.2025.00535

Equal Employment Opportunity Commission, Civil Service Commission, Department of Labor, & Department of Justice. (1978). Uniform guidelines on employee selection procedures. Federal Register, 43(166), 38290–38315.

European Union. (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union.

Fairley, R. E. (2014). Software engineering competency model (SWECOM), Version 1.0. IEEE Computer Society.

Feller, J., & Dakan, R. (2025). AI fluency: A framework for human and AI collaboration. Anthropic.

Google. (2025). Start learning all things AI on the new Google Skills. Google. https://blog.google/products-and-platforms/products/education/google-skills/

IEEE Computer Society. (2024). Guide to the software engineering body of knowledge (SWEBOK®): Version 4.0 (H. Washizaki, Ed.). IEEE Computer Society.

International Organization for Standardization & International Electrotechnical Commission. (2023). ISO/IEC 42001:2023: Information technology, artificial intelligence, management system. ISO.

Lightcast. (2024). Open skills taxonomy. Lightcast.

Long, D., & Magerko, B. (2020). What is AI literacy? Competencies and design considerations. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (pp. 1–16). Association for Computing Machinery. https://doi.org/10.1145/3313831.3376727

Maes, S. H. (2026). Evaluating the efficacy of artificial intelligence in software engineering: A post-February 2026 analysis [Working paper].

Metzger, L. S., & Bender, L. R. (2007). MITRE systems engineering (SE) competency model (Version 1.13E). The MITRE Corporation.

National Association of Colleges and Employers. (2025). Nearly two-thirds of employers use skills-based hiring practices for new entry-level hires. NACE.

National Center for O*NET Development. (n.d.). O*NET content model. O*NET OnLine. Retrieved June 2026, from https://www.onetonline.org

National Institute of Standards and Technology. (2023). Artificial intelligence risk management framework (AI RMF 1.0) (NIST AI 100-1). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.100-1

Ng, D. T. K., Leung, J. K. L., Chu, K. W. S., & Qiao, M. S. (2021). AI literacy: Definition, teaching, evaluation and ethical issues. Proceedings of the Association for Information Science and Technology, 58(1), 504–509. https://doi.org/10.1002/pra2.487

OpenAI. (2025). GDPval: Measuring the performance of our models on real-world tasks. OpenAI. https://openai.com/index/gdpval/

OpenAI. (2026). The AI jobs transition framework: Mapping AI's near-term impact on jobs. OpenAI.

Presland, I., Beasley, R., Gelosh, D., Heisey, M., & Zipes, L. (2018). INCOSE systems engineering competency framework (INCOSE-TP-2018-002-01.0). International Council on Systems Engineering.

Schippmann, J. S., Ash, R. A., Battista, M., Carr, L., Eyde, L. D., Hesketh, B., Kehoe, J., Pearlman, K., Prien, E. P., & Sanchez, J. I. (2000). The practice of competency modeling. Personnel Psychology, 53(3), 703–740. https://doi.org/10.1111/j.1744-6570.2000.tb00220.x

SFIA Foundation. (2024). Skills framework for the information age (SFIA), Version 9. SFIA Foundation.

Society for Industrial and Organizational Psychology. (2018). Principles for the validation and use of personnel selection procedures (5th ed.). American Psychological Association.

Stack Overflow. (2025). 2025 Stack Overflow developer survey. Stack Overflow. https://survey.stackoverflow.co/2025/

Tomlinson, K., Jaffe, S., Wang, W., Counts, S., & Suri, S. (2025). Working with AI: Measuring the applicability of generative AI to occupations. Microsoft Research. https://arxiv.org/abs/2507.07935

Wang, B., Rau, P.-L. P., & Yuan, T. (2023). Measuring user competence in using artificial intelligence: Validity and reliability of artificial intelligence literacy scale. Behaviour & Information Technology, 42(9), 1324–1337. https://doi.org/10.1080/0144929X.2022.2072768

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

World Economic Forum. (2025). The future of jobs report 2025. World Economic Forum.

# Appendices

## Appendix A: Summary of Changes from Version 2.0 to 2.1

Version 2.1 is the most extensive revision since the model's inception, affecting every level of the framework from categories to subskill labels. The table below summarizes the principal changes, together with the primary evidentiary basis for each. It is not exhaustive at the individual-subskill level; the complete current structure is given in Appendix C.

| Change | Description | Primary basis |
|---|---|---|
| AI elevated to a full category | Five AI skills (Fluency, Collaboration, Responsibility, Governance, Engineering) replace the dispersed AI-readiness additions of v2.0. | Landscape survey; AI-literacy literature; SME input |
| Supporting category renamed to Business | The v2.0 Supporting category was renamed Business to better convey that these interpersonal, collaboration, and delivery competencies are broad organizational capabilities, and to read clearly to non-technical stakeholders. | Landscape survey; SME input |
| Technology Stack established as a category | Programming languages and technologies, previously held outside the model in standalone reference appendices (v2.0 Appendix A and Appendix B), were brought into the model as a distinct category with two skills (Programming Languages; Technologies) and 107 subskills. The individual languages and technologies are largely carried forward, not new content; the change is structural. Because they were never part of the model before, every Technology Stack entry is new to the model in v2.1. | Lightcast; O*NET Hot Technologies |
| AI skills fully restructured | The four v2.0 AI-readiness groupings (AI Literacy, AI Evaluation, AI Application, AI Building) and their subskills (e.g., Prompt Writing, Prompt Engineering, AI Output Proofreading, AI Tools and Applications, Model Tuning, Statistical Modeling) were retired and replaced by the five v2.1 AI skills and a refreshed subskill set (e.g., Context Engineering, AI Output Validation, Agent Orchestration, MLOps). | AI Fluency framework; NIST AI RMF; SME input |
| AI governance and responsibility added | New AI Governance (AI Regulation, AI Policy, Human Oversight, AI Risk Management) and AI Responsibility (Algorithmic Fairness, AI Explainability, AI Accountability, AI Ethics) skills were introduced, grounding the model in current regulation and standards. | NIST AI RMF; EU AI Act; ISO/IEC 42001 |
| Subskills renamed for clarity and currency | Numerous technical subskills were renamed, including Release Management to Revision Control, Pen Testing to Penetration Testing, Incident Management to Security Incident Management, Audit Policies to Security Auditing, Error/Exception Handling to Exception Handling, Leveraging APIs to API Integration, Low-level Programming to Embedded Programming, User Accessibility to Accessibility, and Data Generation to Test Data Generation. | SME input; organizational survey |
| Definitions broadened and modernized | Skill definitions were rewritten to reflect current practice, including a broadened Working with Data definition covering statistical and ML modeling and ongoing model tuning, and updated Engineering Operations and Security definitions. | Landscape survey; SWEBOK v4 |
| Architecture skills consolidated | Software Architecture and Systems Architecture merged into Software & Systems Architecture, retaining all subskills. | Organizational survey; SWEBOK v4 |
| Project Management added | New Business-category skill covering delivery and Agile methodologies and management of scope, time, and budget. | Organizational survey (corroborated gap) |

| | | |
|---|---|---|
| Category architecture re-anchored | Five categories (Technical, AI, Problem-Solving, Business, Technology Stack) mapped to authoritative external frameworks. | Landscape survey |
| Technology coverage refreshed | Programming languages and technologies updated to reflect current labor-market demand. | Lightcast; O*NET Hot Technologies |

# Appendix B: Evidence Sources Informing Version 2.1

The Version 2.1 revision drew on multiple, complementary sources of evidence, consistent with the requirement that competency models rest on more than a single method (AERA, APA, & NCME, 2014; Campion et al., 2011; SIOP, 2018):

1. **Authoritative external frameworks.** Government job-analysis data (O*NET), discipline-wide bodies of knowledge (SWEBOK v4, SFIA v9), psychometrically validated competency models (the Great Eight), binding regulation and standards (EU AI Act, ISO/IEC 42001, NIST AI RMF), practitioner frameworks (AI Fluency), and empirical labor-market data (Lightcast).
2. **Primary expert input.** Internal and external subject-matter experts contributed to the naming, bounding, and organization of skills, with particular weight in the AI domain where formal frameworks lag practice.
3. **Organizational confirmation.** A structured feedback survey of Codility's engineering organization tested the category, skill, and subskill structure and surfaced the corroborated refinements adopted in 2.1.
4. **Contextual research.** Peer-reviewed, industry, and frontier-laboratory research on AI's effect on engineering work and on the measurement of AI competence (e.g., Anthropic, 2026a, 2026b; Cui et al., 2025; OpenAI, 2025; Stack Overflow, 2025; Tomlinson et al., 2025; Wang et al., 2023; World Economic Forum, 2025) informed the framing and prioritization of the revision.

# Appendix C: Skill Definitions and Subskills

The following tables present each skill in the model with its definition and constituent subskills, organized by category. For the Technology Stack category, the full catalog of programming languages and technologies is listed in Appendix D.

## *Technical*

| **Configuration Management** | Tracking, controlling, and documenting individual components and assets within IT systems and software to ensure consistency, traceability, and reproducibility throughout their lifecycle (e.g., setting up build systems, makefiles, versioning software components, maintaining a comprehensive record of system configurations). |
|---|---|
| *Subskills* | Revision Control |

| **Engineering Operations** | Planning, delivering, and controlling operations to streamline software development. Applying automation, using continuous integration and delivery (CI/CD), and applying deployment strategies and infrastructure-as-code paradigms while adhering to security and observability practices. |
|---|---|
| *Subskills* | None |

| **Security** | Safeguarding software and systems by adhering to secure coding standards, implementing cryptographic techniques, and ensuring robust infrastructure security measures to protect against potential threats and vulnerabilities. |
|---|---|
| *Subskills* | Cryptography, Infrastructure Security, Secure Coding Standards, Penetration Testing, Security Incident Management, Security Auditing |

| **Software Construction** | Developing software by implementing user interfaces, managing errors, asynchronous tasks, and memory; applying object-oriented principles, algorithms, and APIs; and supporting cross-platform compatibility, scripting, concurrency, event-driven and functional programming, recursion, linear and low-level programming, and user accessibility. |
|---|---|
| *Subskills* | Algorithms, Functional Programming, API Integration, Linear Programming, Embedded Programming, Object-Oriented Programming, Recursion, Resource Management, Responsive Design, Scripting, UI Implementation, Asynchronous Operations, Accessibility, Basic Programming, Code Review, Concurrency, Cross-Platform Development, Data Structures, Exception Handling, Event-Driven Programming |

| **Software Maintenance** | Managing and improving software code continuously to sustain its functionality, performance, and reliability. |
|---|---|
| *Subskills* | Code Optimization, Code Readability, Debugging |

| **Software Requirements** | Collecting and communicating functional and non-functional requirements for a software product and translating them into technical designs while considering user, business, and technical context. |
|---|---|
| *Subskills* | None |

| | |
|---|---|
| **Software Testing** | Applying software testing techniques, levels, and automation to ensure software functionality, reliability, and quality. |
| *Subskills* | Test Data Generation, Quality Assurance, Software Design Review, Software Testing Techniques, Test Automation, Testing Levels |

| | |
|---|---|
| **Software & Systems Architecture** | Designing robust, efficient software and systems by applying architectural principles and fundamental software structures (e.g., design patterns, object-oriented design, microservices, code structure); determining system configurations; analyzing, debugging, and monitoring systems. |
| *Subskills* | System Analysis, System Debugging, System Design, System Monitoring, Design Patterns |

| | |
|---|---|
| **Working with Data** | Examining and interpreting data through querying, visualization, reporting, and statistical and ML modeling to gain insights and make informed decisions, including data preparation, sourcing, exploration, and ongoing model tuning and maintenance. |
| *Subskills* | Data Querying, Data Exploration, Data Preparation, Data Reporting, Data Sourcing, Data Privacy, Data Visualization, Large-Scale Data Analysis |

## *Artificial Intelligence*

| | |
|---|---|
| **AI Fluency** | Working effectively, efficiently, ethically, and safely within emerging modalities of human-AI interaction by understanding what AI is, how it works, what it can and cannot do, and developing the critical awareness to engage with AI systems appropriately. |
| *Subskills* | AI Foundations, AI Failure Modes, Context Engineering, AI Data Literacy, Context Hygiene |

| | |
|---|---|
| **AI Collaboration** | Working effectively with AI as a collaborative partner across the complete interaction loop, from task delegation through output refinement, and supervising autonomous AI agents. |
| *Subskills* | AI Task Delegation, AI Iteration, AI Output Validation, Agent Orchestration |

| | |
|---|---|
| **AI Responsibility** | Practicing the values, principles, and skills required to ensure AI is developed and used fairly, transparently, and accountably, treating ethics as an active competency rather than abstract knowledge. |
| *Subskills* | Algorithmic Fairness, AI Explainability, AI Accountability, AI Ethics |

| | |
|---|---|
| **AI Governance** | Operating within regulatory frameworks, organizational policies, and industry standards governing AI use, ensuring AI readiness includes legal and procedural competence. |
| *Subskills* | AI Regulation, AI Policy, Human Oversight, AI Risk Management |

| | |
|---|---|
| **AI Engineering** | Designing, developing, deploying, and maintaining AI and ML systems, primarily for technical roles but including skills relevant to technical product managers and AI-adjacent engineering roles. |
| *Subskills* | Machine Learning, LLM Development, AI Architecture, MLOps, AI Evaluation, AI Security, AI Cost Optimization |

## Problem-Solving

| | |
|---|---|
| **Systems Thinking** | Understanding how the parts of a system interrelate and how changes in one part affect the whole, taking into account the user's perspective and the broader context in which the system and its components fulfill their intended objectives. |
| *Subskills* | None |

| | |
|---|---|
| **Creative Thinking** | Generating original ideas, finding unique solutions to problems, and combining existing ideas in new ways. |
| *Subskills* | None |

| | |
|---|---|
| **Critical Thinking** | Evaluating information and arguments through reasoned analysis to make well-informed decisions, solve problems, and facilitate learning, applying skepticism, logical reasoning, and the ability to question assumptions and biases. |
| *Subskills* | None |

| | |
|---|---|
| **Computational Thinking** | Formulating and solving problems using strategies common to computer science and programming, breaking down complex problems into smaller components, and expressing solutions in a language that a computer can execute (Wing, 2006). |
| *Subskills* | None |

## *Business*

| | |
|---|---|
| **Teamwork** | Working effectively with others on shared tasks (e.g., co-creating, pair programming), resolving conflicts through facilitation and reconciliation, participating actively in team discussions, assisting other members in addressing problems, emphasizing collective interests, sharing information and knowledge, and acknowledging and advocating for others' contributions. |
| *Subskills* | None |

| | |
|---|---|
| **Communication** | Conveying information, ideas, thoughts, and feelings effectively and clearly through verbal, written, non-verbal, and digital channels; listening actively; expressing oneself coherently; adapting communication style to different audiences; and comprehending and interpreting information from others accurately. |
| *Subskills* | None |

| **Developing Others** | Developing, guiding, teaching, coaching, and mentoring others toward achieving personal and organizational goals, and delivering effective feedback on an ongoing basis to improve performance. |
|---|---|
| *Subskills* | None |

| **Promoting Inclusivity** | Respecting, appreciating, and encouraging contributions and participation from all individuals with diverse backgrounds and cultures, and fostering an inclusive mindset that evaluates perspectives, practices, and products through diverse cultural lenses. |
|---|---|
| *Subskills* | None |

| **Learning and Adapting** | Responding to volatility, uncertainty, complexity, and ambiguity (VUCA) with curiosity and flexibility; seeking new ideas and approaches by applying learning to work; discarding ineffective methods; and adapting plans, priorities, and goals in response to difficult conditions, tight deadlines, obstacles, and setbacks. |
|---|---|
| *Subskills* | None |

| **Taking Ownership** | Taking responsibility for one's decisions and behaviors, holding oneself accountable for meeting work objectives within set timeframes, and acting without waiting for direction. |
|---|---|
| *Subskills* | None |

| **Behaving Ethically** | Exhibiting integrity, trustworthiness, honesty, and fairness in one's decisions and actions, and demonstrating global, social, intellectual, and technological responsibility. |
|---|---|
| *Subskills* | None |

| **Achieving Results** | Setting goals for personal and group accomplishment, monitoring progress toward goal attainment, working to meet or exceed goals, persisting to accomplish tasks, and being outcome-driven rather than output-driven. |
|---|---|
| *Subskills* | None |

| **Domain Expertise** | Applying occupation- or function-specific knowledge required to perform work in a particular domain or industry context (e.g., marketing, customer service, sales, finance, healthcare, legal). Includes the vocabulary, regulations, customer or product knowledge, and workflows that is learned upon entering a specific function. |
|---|---|
| *Subskills* | None |

| **Project Management** | Planning, organizing, and delivering work within real-world constraints: applying delivery and Agile methodologies (e.g., Scrum, Kanban), managing scope, timelines, and budgets, balancing trade-offs between ideal and feasible solutions, and coordinating resources to achieve outcomes. |
|---|---|

| | |
|---|---|
| *Subskills* | None |

## *Technology Stack*

| | |
|---|---|
| **Programming Languages** | Knowledge of formal programming, query, and markup languages used to design, implement, and maintain software systems, including syntax and semantics, idiomatic style, paradigms (e.g., imperative, functional, object-oriented), standard library use, ecosystem and tooling conventions, and the trade-offs that make a given language suitable for a particular problem. |
| *Subskills* | See Appendix D (27 items). |

| | |
|---|---|
| **Technologies** | Knowledge of frameworks, libraries, platforms, runtimes, and tools used to build, deploy, test, secure, and operate software systems, including core concepts and architecture, common usage patterns, configuration, integration with adjacent technologies, and the contexts in which each is appropriately applied. |
| *Subskills* | See Appendix D (80 items). |

# Appendix D: Subskill Definitions

This appendix provides the definition of each subskill in the model, organized by category and skill. The Technology Stack category is excluded here, as its entries are the individual programming languages and technologies catalogued in Appendix E.

## *Technical*

### Configuration Management

| Subskill | Definition |
|---|---|
| **Revision Control** | Applying common revision control notions and patterns (e.g., file revision, history, log, blaming, branching, release, patching, backporting) using systems such as Git, Perforce, Mercurial, or Subversion. |

### Security

| Subskill | Definition |
|---|---|
| **Cryptography** | Applying techniques for securing data and communication (e.g., encryption). |
| **Infrastructure Security** | Applying cloud infrastructure security concepts (e.g., VNets, VPN, AWS WAF, network firewalling). |
| **Secure Coding Standards** | Applying secure coding practices and defensive programming techniques such as threat and risk testing. |
| **Penetration Testing** | Applying ethical hacking techniques to simulate attacks that expose security vulnerabilities in apps, data networks, and other assets. |
| **Security Incident Management** | Identifying, responding to, investigating, and resolving security incidents, and documenting and reviewing incident reports and risk assessments. |
| **Security Auditing** | Checking compliance with legal and regulatory security requirements using security frameworks, forensic tools, and risk assessments. |

### Software Construction

| Subskill | Definition |
|---|---|
| **Algorithms** | Implementing software that runs within time and memory constraints. |
| **Functional Programming** | Building software by composing pure functions, avoiding shared state, mutable data, and side effects in a declarative way. |
| **API Integration** | Consuming and implementing operating system or web APIs and their documentation, and applying common standards (e.g., OpenAPI, POSIX, REST, GraphQL, OAuth, HTTP). |
| **Linear Programming** | Applying linear optimization algorithms and libraries to solve problems with linear objective functions and constraints. |
| **Embedded Programming** | Designing, implementing, and optimizing software code to run on embedded systems (e.g., mobile hardware, automotive systems, wearable devices, industrial controls, networking equipment). |
| **Object-Oriented Programming** | Applying objects, classes, types, encapsulation, inheritance, and interfaces to improve maintainability and reduce complexity. |
| **Recursion** | Designing functions or data structures that are self-referential. |
| **Resource Management** | Managing pointer and reference semantics and memory leaks (e.g., sharing, object ownership, garbage collection, NULL pointer/reference, cyclic structures, memory allocation, copy-on-write). |
| **Responsive Design** | Applying responsive design principles for both desktop and mobile websites or applications. |

| Subskill | Definition |
|---|---|
| **Scripting** | Building scripts in glue languages to integrate and orchestrate existing programs. |
| **UI Implementation** | Implementing user interface designs (e.g., visual elements, interactions, animations, transitions). |
| **Asynchronous Operations** | Implementing code that performs operations without blocking the main execution flow, allowing tasks to run independently and complete in non-sequential order (e.g., callbacks, promises, async/await). |
| **Accessibility** | Applying methods that make websites usable by all people, regardless of disability (e.g., WCAG standards, ARIA, accessibility patterns). |
| **Basic Programming** | Applying fundamental coding practices and techniques (e.g., assignment, arithmetic expressions, arrays, strings, variables, loops, conditions, input/output, functions). |
| **Code Review** | Examining source code changes systematically to evaluate implementation, identify defects, and ensure adherence to standards and best practices, while providing constructive feedback and facilitating knowledge sharing among team members. |
| **Concurrency** | Applying common concurrency patterns (e.g., synchronization primitives, inter-process communication, threads, shared memory) and recognizing common concurrency issues (e.g., deadlocks, livelocks, starvation, race conditions, distributed systems). |
| **Cross-Platform Development** | Developing software that runs consistently across multiple platforms, operating systems, devices, or environments (e.g., iOS and Android, Windows/macOS/Linux, web and desktop). |
| **Data Structures** | Applying data structures across complexity levels (e.g., set, list, queue, stack, dictionary, trees, graphs, priority queue, tabular data). |
| **Exception Handling** | Applying common error and exception handling patterns and exception safety concepts. |
| **Event-Driven Programming** | Designing and debugging event-driven systems (e.g., event sourcing, event bus, event loop) and recognizing common event-driven programming issues (e.g., at-least-once and at-most-once delivery, event taxonomy, eventual consistency). |

### Software Maintenance

| Subskill | Definition |
|---|---|
| **Code Optimization** | Identifying and removing bottlenecks through re-coding and re-architecting to optimize for performance, scaling, memory usage, or other criteria. |
| **Code Readability** | Producing software that is easy for other developers to understand, read, and reuse (e.g., appropriate use of comments, documentation, code structure, coding conventions, and self-explanatory naming). |
| **Debugging** | Identifying, diagnosing, reproducing, and removing bugs in code. |

### Software Testing

| Subskill | Definition |
|---|---|
| **Test Data Generation** | Generating representative sample data for tests, including edge cases and boundary conditions. |
| **Quality Assurance** | Reviewing, monitoring, and auditing software products and development activities to verify quality criteria and standards are met. |
| **Software Design Review** | Reviewing and auditing software design, mockups, documentation, architecture, and requirements. |
| **Software Testing Techniques** | Implementing software testing techniques (e.g., boundary analysis, use case matrix, equivalence partitioning, use case testing, white/black box testing, feature testing). |

| Subskill | Definition |
| --- | --- |
| **Test Automation** | Designing, writing, and implementing automated test scenarios including test prerequisites, steps, and results. |
| **Testing Levels** | Implementing software testing levels (e.g., unit testing, integration testing, system testing, performance testing). |

### Software & Systems Architecture

| Subskill | Definition |
| --- | --- |
| **System Analysis** | Determining how a system should work and how changes in conditions, operations, and the environment will affect outcomes. |
| **System Debugging** | Identifying, diagnosing, reproducing, and removing bugs in distributed systems using strategies such as heuristics (e.g., USE), tests, and model checks. |
| **System Design** | Applying availability, consistency models, cross-system communication, scalability, single points of failure, observability, and related concepts (e.g., microservices). |
| **System Monitoring** | Applying methods for tracking, reviewing, and regulating system performance (e.g., system configuration metadata changes). |
| **Design Patterns** | Applying general, reusable solutions to common problems in software design. |

### Working with Data

| Subskill | Definition |
| --- | --- |
| **Data Querying** | Applying searching and sorting techniques (e.g., joins, views) within databases via a query language (e.g., SQL, Realm, SQLite). |
| **Data Exploration** | Applying data exploration concepts and techniques to identify data characteristics, trends, and problems (e.g., computing and interpreting central tendencies, reviewing distributions, finding missing values). |
| **Data Preparation** | Applying processes for fetching, selecting, cleaning, manipulating, and transforming data to prepare for analysis. |
| **Data Reporting** | Presenting data to stakeholders and other audiences by translating insights and implications from analyses at an appropriate level. |
| **Data Sourcing** | Applying processes for extracting and finding data from internal and external sources. |
| **Data Privacy** | Upholding data privacy principles and practices to ensure the secure and ethical handling of sensitive information (e.g., consent management, data minimization, anonymization, regulatory compliance). |
| **Data Visualization** | Producing graphic representations of data using charts, graphs, and maps. |
| **Large-Scale Data Analysis** | Analyzing and interpreting large datasets using data analytics tools, statistical analysis, and data visualization techniques to glean insights. |

## *Artificial Intelligence*

### AI Fluency

| Subskill | Definition |
| --- | --- |
| **AI Foundations** | Recognizing AI systems, differentiating AI types, understanding capability boundaries, and identifying appropriate applications. |

| Subskill | Definition |
|---|---|
| **AI Failure Modes** | Recognizing how AI systems fail, including hallucinations, automation bias, uncertainty, and recurring failure patterns. |
| **Context Engineering** | Structuring clear requests to AI by providing the information a task needs to be solved, including instructions, context, output format, and technical specifications. |
| **AI Data Literacy** | Understanding how data shapes AI behavior, including data quality assessment, privacy implications, and data pipeline design. |
| **Context Hygiene** | Managing the AI context window to maintain agent performance, accuracy, and efficiency by minimizing context pollution, curating the set of available tools, separating tasks across context windows, and pushing state out of chat history when appropriate. |

### AI Collaboration

| Subskill | Definition |
|---|---|
| **AI Task Delegation** | Evaluating which tasks are appropriate for AI versus human execution, including task-AI fit analysis, task decomposition for AI, risk-appropriate delegation, AI tool selection, and workflow redesign. |
| **AI Iteration** | Applying the prompt-verify-refine loop to progressively improve AI outputs, including progressive refinement, few-shot guidance, structured reasoning elicitation, and advanced prompt architecture. |
| **AI Output Validation** | Cross-referencing claims, checking logical consistency, applying domain expertise, and using systematic evaluation frameworks to verify AI output. |
| **Agent Orchestration** | Designing and coordinating multi-agent systems where specialized AI agents pass structured outputs to one another to complete complex workflows, including agent-to-agent protocols, task decomposition across agents, agent role design, progress monitoring, failure recovery, and human-in-the-loop checkpoints. |

### AI Responsibility

| Subskill | Definition |
|---|---|
| **Algorithmic Fairness** | Recognizing bias sources, assessing differential impacts on groups, applying fairness criteria, and implementing bias mitigation techniques. |
| **AI Explainability** | Communicating AI decisions and limitations to stakeholders and applying explainability techniques (e.g., LIME, SHAP, attention visualization). |
| **AI Accountability** | Taking ownership of AI-assisted decisions, preserving human agency, and exercising judgment about when to escalate. |
| **AI Ethics** | Identifying ethical concerns in AI use, analyzing stakeholder impacts, navigating values trade-offs, and applying ethical frameworks. |

### AI Governance

| Subskill | Definition |
|---|---|
| **AI Regulation** | Understanding the AI regulation landscape (e.g., EU AI Act), risk classifications, organizational obligations, and jurisdictional considerations. |
| **AI Policy** | Following organizational AI policies and data handling rules and contributing to policy development. |
| **Human Oversight** | Applying the human oversight competencies required for high-risk AI systems (e.g., EU AI Act Article 14), including system understanding, anomaly detection, intervention authority, and oversight documentation. |

| Subskill | Definition |
| --- | --- |
| **AI Risk Management** | Identifying AI risks, reporting incidents, maintaining technical documentation, and conducting conformity assessments. |

### AI Engineering

| Subskill | Definition |
| --- | --- |
| **Machine Learning** | Applying algorithm selection based on problem type and data characteristics, model training, evaluation metrics interpretation, and feature engineering. |
| **LLM Development** | Building LLM-based applications through API integration, system prompt engineering, retrieval-augmented generation (RAG), and fine-tuning. |
| **AI Architecture** | Designing end-to-end AI solutions, agent and multi-agent systems, and scalable architectures. |
| **MLOps** | Deploying, monitoring, and maintaining models in production, and operating CI/CD pipelines for ML. |
| **AI Evaluation** | Applying model testing, evaluation framework design, LLM evaluation, and security testing for AI vulnerabilities. |
| **AI Security** | Defending AI systems against adversarial threats, including prompt injection (direct and indirect), jailbreaks, data poisoning, model extraction, and tool misuse, by applying input validation, prompt scaffolding, output filtering, privilege limitation, guardrail systems, and red-teaming. |
| **AI Cost Optimization** | Optimizing the cost and efficiency of AI systems in production by selecting appropriately sized models for each task, applying prompt and context compression, using prompt caching and batch APIs, monitoring token consumption, and balancing latency, accuracy, and unit economics. |

# Appendix E: Programming Languages and Technologies Covered

**Programming Languages (27)**

C, JavaScript, Kotlin, Lua, Objective-C, Oracle SQL, Pascal, Perl, PHP, Python, R, C#, Ruby, Rust, Scala, Solidity, SQL, Swift, TypeScript, Visual Basic, C++, COBOL, Dart, Elixir, Go, Haskell, Java

**Technologies (80)**

.NET, Azure, Bash, Blockchain, Chef, CSS, Cucumber, DB2, Django, Docker, DynamoDB, Android, Elasticsearch, Entity Framework, Excel, Express.js, FastAPI, Flask, Flutter, Git, Angular, GitLab CI, Google Cloud Platform, GraphQL, Hibernate, HTML, iOS, Java Streams, Jenkins, Jest, jQuery, JSON, JUnit, Ansible, Kubernetes, LangChain, LangGraph, Laravel, Linux, Mocha, MongoDB, MySQL, next.js, Node.js, Apache Camel, NoSQL, Open API, oracle, Pandas, Playwright, Powerpoint, PowerShell, Puppeteer, PySpark, PyTest, Apache Spark, React, React Native, Redux, Regular Expressions, Robot, RSpec, Ruby on Rails, Salesforce, SAP, Selenium, ASP.NET Core, Serverless Framework, Shell, Spring, Spring Boot, Symfony, Tableau, Terraform, Vue.js, XML, xUnit, AWS, YAML, AWS Lambda

*Note. This catalog lists languages and technologies at a general level and is refreshed periodically to reflect current labor-market demand. For supported versions, compilers, runtimes, and databases, refer to Codility's technology support documentation.*