DEAPening Employer Academic Partnerships

# DEAP Competencies: Cross-Disciplinary Skills

Version 1, 12/23/2022

**Prepared By Dr. Marisa Exter, Dr. Mihaela Sabin, Dr. Shamila Janakiraman, Deepti Tagare, MS, Ankita Kotangale, MS, Dr. Suzhen Duan**

National Science Foundation

*This material is based upon work supported by the National Science Foundation under Grant No. 2111097*

**Table of Contents**

# Introduction

As part of our initiative to determine competencies important to computing professionals, we interviewed 32 individuals in computing related roles who work in a range of industries and have varied roles, responsibilities, and professional and educational backgrounds. We also reviewed 52 articles that included data from computing professionals related to skills, knowledge, and/or dispositions required on the job (with publication dates from 2010-2021). These data sources were analyzed using a thematic analysis approach adapted from the Constant Comparative Method for Naturalistic Inquiry (Lincoln & Guba, 1985).

This document is the first draft report of the findings related to Cross-Disciplinary Skills from across data sources. In each section, you will find:

- A general description of the category of cross-disciplinary skills.
- A nested list of related cross-disciplinary skills. Under each, you will find a description and one to three illustrative quotes. Note that the description is based on themes in the data – it is *not* a definition or set of criteria for assessing these skills.

# Collaboration and Teamwork

Interview data and literature reviewed highlight the importance of collaboration and teamwork skills. All interview participants work with others on a regular basis, including working within a team as well as one-on-one interactions and interactions with other stakeholders. When asked what they look for when hiring, managers and others who take part in the hiring process indicate that this is something they ask about in detail, by asking probing questions about prior experience working in teams. Some even require new hires to demonstrate their ability to collaborate through interactive activities during interviews.

This section describes sets of competencies that are highly important to successful collaboration on the job. Both interviews and literature recommend that learning experiences be designed to allow students to develop and practice these competencies.

## Building Relationships

### Building Trust

**Description**

- Building a relationship of mutual trust with team-members and other stakeholders
- Engaging in "small talk" (talking about unimportant topics) may encourage social connections

**Quotes**

> You've built that trust and teamwork with them that they know, 'hey, he's asking me for this right now. You know, it's probably going to be pretty important'… I'm going to do something for them when they need it. So, you're building up a trust amongst your members within our own group. (Interview)

> The first one that I think is really helpful is getting people to trust me, especially with the people who work for me. They need to be able to tell me bad news. Like if they if they are behind on a project or something isn't turning out like they thought it was going to… I really try to create an atmosphere where you know they can come to me with whatever the issue is. (Interview, Manager)

> Pleasantries, introductions can go along way. A very nice how are you doing this morning, going into all communications where you're trying to both impress the person and make them trust you. In addition to your typical business professionalism. On top of that, but I guess underneath we're all humans, so go for that first, then go for the professional part. (Interview)

### Conflict Management

**Description**

- Collaborating to identify win-win solutions
- Creating an atmosphere where debates are not considered conflicts
- Utilizing conflict management strategies

**Quotes**

> Software engineers are capable of resolving conflicts constructively. (Literature – Sedelmaier & Landes, 2014)

### Building Mutually Beneficial Relationships

**Description**

- Creating a relationship that is positive and beneficial from both parties' perspectives
- Cultivating contacts within own company or elsewhere
- Remaining in touch with contacts
- Networking

**Quotes**

> Learn to depend on other people. You don't have to know everything. Find people who do know it. And cultivate those relationships. Yeah, I have quite a few people within the organization here at Purdue that know things about different aspects that I don't know, but I know I can trust them to go talk. I know enough about what they know that I can communicate with them intelligently. It is important that you know enough about a topic to communicate with an expert about it, but don't try to be that expert if that's not your forte. You've got your areas where you are an expert. (Interview)

## Maintaining Inter-organizational Relationships

**Description**

- Being aware of developments in other organizations
- Communicating effectively with other organizations

**Quotes**

> Communicate across organizational boundaries (consult others and keep them informed of developments that may affect them.) (Literature – Ruff & Carter, 2015)

## Coordinating with Others

**Description**

- Coordinating and prioritization work among team members
- Divide and conquer (see strategic Proceses→ Problem Solving → Collaborative Problem Solving for more collaborative approaches)
- Ensuring your work does not negatively affect others' work
- Communicating in a way that takes into account the organizational culture and needs (e.g., if working with a geographically dispersed team or a team including remote workers)

**Quotes**

> I've definitely been in group projects before in college where people are kind of passing blame in terms of like who's picking up the most workload and that kind of thing. So that's probably pretty valuable from like an educational standpoint, being able to divide and conquer in terms of like having like a big task in front of you and giving the roles in between. (Interview)

> Learning how to optimize in a group setting so you don't break people's stuff or delay people was kind of a soft mixed technical skill that we had to carefully manage. I get a little scared doing Unix where I'm just like writing this one big command and then I'd blow out everybody's code. (Interview)

> [It's] a learned skill, how to get something out of someone. We have procedures we go through, tickets, systems and things like that. 'Hey, I need this done. I need this firewall port open.' There are times where you kind of need something sooner than what the normal traditional path takes, and if you've built that relationship in that team, work with others. (Interview)

## Cross-functional/Interdisciplinary Collaboration

**Description**

- Experiencing working in inter-disciplinary teams
- Recognizing lack of shared knowledge or perspectives that can contribute to failure

**Quotes**

> [When hiring] if I saw somebody that had any kind of cross training like any kind of team projects where maybe they had to work with somebody from a marketing perspective and somebody from an operations perspective. That kind of experience, really caught my attention, because being able to work on a team is one of the really crucial things (Interview, Manager).
>
> So in the past, if they basically completely the software engineers completely rewritten whatever it was… to the point that the data scientist couldn't navigate it…Alright, it's one thing if you have to like take it and add monitoring and add some error catching and whatever …. But the level of transformation was so substantial that. The data scientists couldn't make heads or tails of the codebase. (Interview)

## Knowledge Sharing

**Description**

- Sharing knowledge transparently with stakeholders (e.g., colleagues, clients, users)

**Quotes**

> Development teams have realized that it is helpful to gain an insight into the linked projects at their distributed development centers. This cohesiveness has enhanced the sharing of software development knowledge among development teams, which in turn has enriched each project's competency to deal with system requirements. Hence, the use of IT facilitates knowledge sharing and changes the way project teams view themselves as integrated teams. (Literature – Chugh et al., 2020)
>
> Software engineers are capable of presenting their ideas and issues from their own area of expertise to others. (Literature - Sedelmaier & Landes, 2014)

## Seeking, Giving & Receiving Constructive Feedback

**Description**

- Providing detailed, specific, and direct constructive feedback or critique
- Being respectful when giving feedback
- Engaging in giving and receiving feedback as part of a design process

This may occur informally or through formal events (e.g., code reviews – see also *Computing Skills→Software Development→Code Review*)

**Quotes**

> A code review without attacking somebody is intelligent or without using words like stupid or dumb. You shouldn't be saying like this is a dumb way to do this or like this looks stupid. This is inefficient. So, we were having to navigate these like frustratingly basic human interaction. I like to give specific and direct feedback but also try and be kind about it. I think that works. People say they like it. I tend to give extremely detailed feedback. (Interview)
>
> I think, getting feedback on your ideas and also getting ideas from other people is an important step in the problem-solving process. You want to go to the people that are not going to have relevant experience and you're going to want to get their ideas, have them say like okay yeah, this is an interesting idea, but I don't think it's going to work for this reason, or I do like this idea. So, getting other people to actually look at your ideas can be really, really helpful, especially for larger problems. So yeah collaboration. (Interview)
>
> Because critique is an art and we want to see how good they've gotten at that art. Can you give feedback in a way that is you know kind but also addresses the issue technically? (Interview)

## Serving as a Subject Matter Expert

### Description

- Using in-depth knowledge about something to be able to solve large problems
- Guiding others in a specific area of knowledge and skills

### Quotes

> I think teaching and grading is a really good responsibility to put on people. I ended up TAing when I was doing my masters and it forced me to look critically at the answers being given by the people I was grading, which was super helpful. when I had to teach people stuff, I had to know it, like, not just know it well enough to pass a test, but know it to sit there for an hour and answer questions from people who are completely lost. Right? And how do I describe this in a different way? And so I do think to the extent that people can be get jobs as TAs or helpers for classes, that is a really good role for them. (Interview)
>
> Point of view in terms of like the data sciences versus software engineers come the language like the actual language barrier is a little bit difficult. When you make a decision to use a certain solution, there's a bunch of kind of like implicit things that an expert would know that somebody outside wouldn't know. (Interview)

# Communication Skills

Both literature and interviews stress the importance of communication, including in-person and online communication with individuals and groups; and creating written and visual elements, typically part of technical writing in the form of design documents, bug reports, etc., as well as less formal writing for interpersonal communication, (e.g., email, instant messages).

Both sources indicate that there is a concern that students are not adequately prepared for the quality of communication required to perform well on the job, and indicated that institutions of higher education need to provide further opportunity to develop and practice these competencies.

This section will describe the most common communication-related competencies. Note that one of the primary sets is “Tailoring the message,” which relates to communicating appropriately for different stakeholders in different circumstances.

## Non-Verbal Communication

**Description**

- Communicating without words, e.g., through facial expressions, eye contact (or lack thereof), body language, gestures, or physical distance

**Quotes**

> Adjust communication based on non-verbal reactions of audience (Literature – Ruff & Carter, 2015)
>
> Manage one's own non-verbal communication to avoid sending inappropriate messages (Literature - Ruff & Carter, 2015)

## Presentation Skills

**Description**

NOTE: while items in this group refer to other communication competencies, they are applied to a formal presentation to a group

- Speaking clearly, concisely, and confidently *(see Clear Verbal Communication)*
- Presenting material in a compelling manner and at a level appropriate for all members of the audience *(See Tailoring the Message)*
- Utilizing non-verbal language appropriately while presenting (see *Non-Verbal Communication)*
- Arranging content in a well-organized manner (e.g., logically ordered and chunked to promote understanding and retention)

**Quotes**

> While presenting it's very important that you are making eye contact and talking through to your audience even though you're just talking to a camera or computer screen. You have to be able to present yourself in a way that not just the person that you're talking to directly understands you. You also want to be able to make sure that the rest of your audience understands you as well.
> Rarely will you be in a position to where you're presenting to one group of people in one specific area. They'll be usually several departments, and they all have their own concentrations. There will be business executives. You'll have sales executives. You'll have the management and then you'll have lower-level administrators. You have to be able to present whatever issue or whatever it is that you're trying to you convey. Make people aware of in a way that everyone can understand. *(Interview)*

## Succinct Communication

**Description**

- Expressing thoughts or explanations things in a concise way

**Quotes**

> One of the biggest challenges that engineers have is communicating across organizations and with people in different roles, engineers have a hard time being succinct. I used to coach people where I'd say, 'Tell people the punchline first. That is a cross organizational challenge for a lot of engineers it's really hard for engineers to speak at a higher, more abstract level. They like to get into details. (Interview)

# Tailoring the Message

## Appropriately Communicate What People Don't Want to Hear

### Description

- Communicating negative messages (e.g., that you can't complete a feature on time or will not use a solution preferred by a stakeholder) utilizing techniques such as presenting concerns with confidence, reframing a message, or offering alternative solutions

### Quotes

> The ability to breakdown instructions into simple step by step processes, especially when sometimes your customer may be afraid of the technology that they're using. And sometimes the ability to just be able to say no, you can't always say yes to everything. You sometimes have to say no and hopefully my goal is that it's 'no, but what if we try this?,' so that you can provide them an alternative. (Interview)

> It definitely takes being assertive and confident when you're in a role where nobody has the same experience as you and you need to communicate an issue to leadership. I mean, the issues that we're dealing with from the perspective an organization, they don't see them as like potential revenue sources or anything. They think about them in terms of liability and cost. And so, you need to communicate or things that upper management does not want to hear about. (Interview)

## Persuasive Communication

### Description

- Representing one's own opinion convincingly and confidently
- Knowing what not to say and thinking things through before saying them
- Recognizing and heading off a potential disagreement and knowing when to end a conversation
- Managing people's expectations

### Quotes

> Being able to understand what they want [and] being able to explain to them what you bring to the table… There's a lot of give and take in that communication, because they have a specific set of ideas and [you need to] make them understand what you're trying to say without making them feel threatened by it. A lot of times if you're trying to maybe bring in a new system, there's already a fear of people. Maybe they were going to lose some of the things that they were doing. Maybe they would lose their job. You know, bringing them into a place where you can communicate with them and set them free of those fears and let them know that you're here to help. (Interview)

> It definitely takes being assertive and confident when you're in a role where nobody has the same experience as you and you need to communicate an issue to leadership and so on. (Interview)

## Predict an Audience's Questions

**Description**

- Being prepared to answer your audience's questions in terms that make sense to them
- This can ensure that stakeholders believe that you know what you are talking about

**Quotes**

> You're trying to predict what questions the audience is likely to have. And make sure that you either are answering those questions in the material you're presenting, or at least you have the answers and you're anticipating the questions, so that you don't stand there like a deer in the headlights when some question comes up that you hadn't thought about that. That obviously can destroy one's credibility very quickly.

## Tailor Language to the Audience

**Description**

- Describing concepts at the right level of detail in terms that each stakeholder understands
- Recognizing that your audience do not think the way you do (e.g., "like an engineer" or "like a UX designer")
- Being a "bridge" by translating between different stakeholders (e.g., those with detailed technical knowledge and those with business knowledge)

**Quotes**

> There's an under population of individuals that can really take technical terms and translate that into business terms. We could have a conversation about something highly technical about DHCP/TCIP, TCIP and IP but for someone standing there, they could be the CEO of a company, it's going to sound like we're speaking a totally different language. To take those concepts and ideas and translate them into something that makes sense from a business standpoint, I think that skill is very lacking. (Interview)
>
> Being able to look at everything that's going on and what you're doing, and not only be able to communicate exactly but also communicate it in a way that is less technical. We need to be able to take our very technical reports and things like that and bring them to a point where they're very understandable by somebody that may not know all the acronyms and things like that. (Interview)

## Understanding How a Message will be Perceived

**Description**

- Thinking from other's perspectives (particularly people different than oneself)

**Quotes**

> Think proactively about how the message will be perceived and present the message. What is it that the audience wants to learn? What point are you trying to make with the audience. Try to

> do the presentations or documentation in a way that gets the message across successfully to the audience in a way that they'll understand. Working with a vendor, working with customers, working with my own management, working with other parts of the organization. The common thread is, what are those people's motivations? What is it that they're trying to accomplish? How do they perceive us, me, and how do we present the information in a way that's perceived as useful and positive and compelling to them? Really do try to put a lot of energy into thinking about what is it that the other party is trying to accomplish here. (Interview)

# Verbal Communication

## Active Listening

**Description**

- Being fully present and engaged when someone is talking to you
- Making a conscious effort to hear, understand, retain, and analyze what someone else is saying
- Asking questions rather than asserting opinions

**Quotes**

> Listen actively (e.g., ask clarifying questions, admit to gaps in understanding). Discern when to ask questions rather than to assert an opinion. Discern when to keep silent rather than to speak. (Literature – Ruff & Carter, 2015)

## Asking Insightful Questions

**Description**

- Asking informed questions to solicit information
- Asking probing questions to determine others' level of understanding

**Quotes**

> But as engineers you should understand what other questions to ask. Those are very important things because if you don't understand what you're going to build, your not going to build it well. So it's extremely important to understand the requirements and ask the right questions because initially you don't have any clue or any idea what that project is or what that feature is? Who is asking that feature? Why we are developing this feature? So with every product or every feature that we develop, we ask those questions to have a an understanding of what the end user wants. If there is a project manager as external stakeholders, we need to ask them what this really is. That is part of the requirement gathering phase. (Interview)

## Clear Oral Communication

**Description**

- Expressing oneself clearly (e.g., good pacing, tone, enunciation, etc.)
- Using English fluently

#### Quotes

> Public speaking, as much as people dislike this type of course, especially undergrad level, it is very important because if you work in any company, and interact with your subordinates or your management or your executive team, you have to be able to present yourself in respectful manner and you have to be able to project what it is that you're trying to say. If you kind of get hung up and stumble upon your words no one will really understand you and no one will really give respect or take your opinion because you just simply didn't present it well. You may have had a good argument, but if you can't present that argument and solution for that problem, you'll just simply be ignored and kind of just shoved aside until it becomes a much bigger problem. (Interview)

### Communicating About Your Work

#### Description

- Being able to describe and explain your own current or prior work experience (including technical aspects, process, approach, etc.)

> [When interviewing] we ask them to talk to us about their experience. Tell us about a project you've written with this technology and we dive in there and we learn how they explain their own work, so you get a really good idea of how they. How they communicate, how they built something. (Interview)

## Written Communication

### Technical Writing

#### Description

- Writing various types of technical documentation (e.g., reports, requirements, design documents, user stories, bug reports, test plans, change logs, etc.)
- Recording technical processes or procedures
- Technical writing is clear, concise, and targeted at a particular audience (See also *Tailoring the Message* and *Succinct Communication*)
- Technical writing allows for replicability

#### Quotes

> Another form of written communication are design documents. It is mostly how the software is going to get designed to solve that particular requirement. Design is a software design, or you can call it as a software architecture. What are the different components of that software? You have to have like 1000 feet view of that particular architecture of your software. And then constantly you have to zoom in into that particular component and then zoom out and then zoom in constantly to get the big picture as well as the detail level of it. It comes with experience. Just write it down. You know what the components were. If we are designing, adding this particular feature, how this particular feature will impact which component? Just write it down in plain words, simple words. (Interview)

> Our builds are used and consumed by like all sorts of developers in China and in Boston or you know anywhere. So it's super important if our process changes that we get those spelled out correctly, and understandably. (Interview)
>
> You have to be able to write in a way that you know somebody who's never been able to or who's never been working in that position, you want them to be able to walk in, read the document and know exactly what to do following the procedure that was given to them, and then ultimately meet the policy criteria that was set out. And if you can do that, then you're certainly meeting the requirements for good technical documentation. (Interview)

## Writing Appropriately for Medium & Content

**Description**

- Style should be appropriate to the medium (e.g., email, instant message, blog) as well as the content

**Quote**

> I mean there are there are good ways to write an email to get someone to help you out, and there are very bad ways to write an email, you know. (Interview)
>
> A subset of written communication is also being cognizant of how tone does and does not get communicated. Because if we've known each other for 10 years and I say OK, that may not mean anything. But if we've known each other for five minutes and I send the curt little OK, with a period on the end of it, you could think, what does that mean? Why did he say that? So it is just being aware of how the same words to different audiences can you mean different things. (Interview)

## Following Writing Conventions for Effective Communication

**Description**

- Writing clearly and concisely
- Using correct spelling and grammar
- Following style conventions

**Quotes**

> Writing clear, concise, and effective memos, reports, and documentation; use correct spelling and grammar most of the time; use structure and formatting to enable fast reading (Literature

## Communicating through Collaborative Software

**Description**

- Utilizing collaborative features of software (e.g., google docs, wikis, slack, github, etc.) to enable effective and efficient interactions (e.g., communication, collaborative writing, multi-user version control, etc.)

**Quotes**

> Engineers review your design doc and see if they can get in, catch anything. If they don't understand things there we have comments. People respond to it and ask for more explanation. If they haven't understood and if you're not clear, then they will ask questions. So it's a good interactive session. We use Google Docs a lot. (Interview)
>
> Slack isn't really documentation… but it is maintaining the line of communication between our team and like the Revit developers or other teams in the future. (Interview)
>
> And then also important is written communication style has to do with our Git workflow. Writing up your merge requests in a way that summarizes your changes and kind of goes over the implications of your change. (Interview)

## Visual Communication

**Description**

- Communicating through graphical depictions
- May be through a formal visual language (e.g., UML) or other standard form (e.g., wireframes) or through an idiosyncratic form (e.g., while whiteboarding, an individual or group may come up with their own short-hand visual language)
- May be done via "whiteboarding" (in person or using a tool), through software-generated diagrams, video, or other media

**Quotes**

> I think probably the biggest one is going to be those mockups. That's generally how in the front end side we communicate our ideas. Because you know words like saying I would have put a box here when the user does this is one thing, but. When you say that people imagine different things, so you need to have like a mockup or a visual image that really represents the changes that you're proposing so when it comes to like communicating ideas generally mockups is that visual language. (Interview)

# Lifelong Learning Skills

Lifelong learners engage in learning for multiple reasons, including to keep up to date with the field; to increase their own skillset in preparation for potential future roles and opportunities; and to respond to immediate project needs (e.g., learning a new tool, technology, programming language, or how to use a particular feature or correct a particular type of error in a just-in-time basis). While they may learn from formal courses or certifications, much learning takes place through self-directed learning.

Life-long learning and continued professional development literature recommends that time spent in formal education be focused on preparing students for lifelong, self-directed, self-determined learning [CITE]. This alleviates the burden of attempting to teach everything a student might need to know for their career – which is not possible in a constantly changing field such as computing.

This section will describe strategies, resources, and practices used by competent life-long learners.

## Self-directed Learning

### Description

- Self-directed learning takes place when an individual determines what and how to learn. Therefore, self-directed learning is not planned or led by an instructor.
- Self-directed learning can be used both to keep up to date, and to address an immediate work need through just-in-time learning
- Self-directed learners may utilize any learning strategy and resource that meets their needs
- Self-directed learners may seek to apply theory they have learned to practice

### Quotes

> So, that's a meta skill of like actually learning how to learn, which I feel like I got out of my four year degree, but I think it's super important of knowing how to keep on top of new things. How to determine what you're going to invest your time in. (Interview)
>
> Paying attention to what your customers are asking you in questions, so they'll want to do new things. And so by paying attention to the questions they're asking, that'll kind of direct what do we need to expand our learning because we have to support that. (Interview)
>
> Recognize one's own communication weakness and improve one's own communication skills. (Interview)
>
> As a compiler engineer, you have to have knowledge of lot many things over here …[the] deep learning machine learning domain. [I am a] software engineer, computer science graduate, and sometimes we don't have much in depth knowledge of these fields, deep learning or machine learning. So we we learn as we go, you know. We learn about the domain, how people are using, how researchers are using it, how the end users are will be using it. We need to understand certain aspects of it, you know, to have some knowledge about it. We may not be as expert as a machine learning engineer, which is a separate role, you know. (Interview)

## Learning Strategies

### Leveraging Prior Knowledge

#### Description

- Recognizing patterns to what you know
- Finding similarities between what you already know and what you need to learn
- Recognizing what is different from what you already know, and what you might need to learn or try

#### Quotes

> So for me actually moving to this job, I never used React which is a JavaScript framework but I've used other JavaScript frameworks that are similar to it. So the way I started there was. One looking at code. It's still within the company and how we're using it. And trying to find the similarities from what I had already done, so what's similar in this single page application. Or a JavaScript framework what's similar to it to Vue, like I've done Vue before so you know what similarities are there? What patterns can I recognize in the code here? The other side is I literally went and found a Udemy course that talked through, literally walked me through creating from scratch a React NestJS application so that I could once again find the things that

> are familiar to me and find what's different so that I can assimilate all of that into how I approach developing in this new technology. (Interview)

## Learning From and With Others

### Description

There are multiple ways to learn through interacting with others. Note that "subject-matter experts" may or may not be senior to the learner. The distinction is that those individuals are more skilled or knowledgeable about an area the learner wishes to learn or improve in. Learning from others is not only useful for gaining specific knowledge or skills, but also provides exposure to their thought processes and ways of approaching tasks or problems.

- Asking questions of subject-matter experts
- Collaborating with subject-matter experts (e.g., thinking through a problem together)
- Learning through observation
- Learning by teaching

### Quotes

> I would grab someone who had been working [on this for] awhile and I would pair with them for an hour. I just say, 'hey, I'm gonna work on this problem. Can you come alongside me and help unblock me on the things I don't understand?' and so there's some people side to it too. (Interview)
>
> It also just helps… me to bounce ideas off other people. So if I'm trying to understanding something, then maybe I'll pull in a colleague and ask them if they already know about it, and how they how they went about doing it. Or, if they don't know about it, I will try and teach them so that way, me teaching them is helping me solidify my knowledge of it. (Interview)
>
> It's equally important to read other design docs as well, and you learn a lot from that as well, because that's how I learned it. Basically, reading other people's work and how they are thinking through and how they are structuring their document, and what is the process, how they're thinking through this problem. (Interview)

## Learning from Experience

### Description

Computing professionals typically learn through "getting their hands dirty". A set of intertwined learning strategies include:

- Tinkering or "playing around", typically by beginning with an example (such as sample code) and making changes to see what they do. This can be done as a step on the way to implementing a solution (such as writing code that does something the learner has not done before), but is also often done by life-long learners for no particular current need (e.g., through hobby projects)
- Learning from failure (including both one's own and others' failure) can be at least as powerful as learning from success.

### Quotes

> There's a real benefit to taking functioning code like copy pasting functioning code from somewhere else and then playing around with it. And making it your own and figuring out like

> how when you tweak certain lines or insert something or remove something like, how does the behavior change. How does the runtime change? I think one thing that I've done with success in the past is just like taking up a program or relatively simple application that I've written, just try to re implement it in a new language. I'm just trying to figure out what is the conversion, what's the mental model? I know C++. How does that look in Python, all the strings stuff gets a lot easier, but I have less control over data types and finding those kinds of trade offs when you're operating a new language. (Interview)
>
> There are people in and I've worked with three of them in my life who are ridiculously good at being dropped into a new code base, a new language, a new system and orienting themselves. They're all tinkerers. One of these guys ran out of gas one day mowing his lawn, so he hooked his propane tank from his grill into the carburetor on the lawn mower. I wouldn't know how to do that like he was just a mechanical tinkerer. He was an electrical tinkerer. So you could drop him into like a new system and he would figure it out. And he had so much experience doing it across so many disciplines.. (Interview)
>
> You learn from failure and you learn from success, but the failure teaches you just as much as or more. (Interview)

## Learning by Writing

**Description**

- Learning by taking notes or writing documentation.
- Some individuals find that the act of writing helps to structure their thoughts, synthesize new knowledge, or even gain a better understanding of a problem or prospective solution

**Quotes**

> If you take the time to like, write out how [the] problem is, I think you'll really understand the solution to it. (Interview)

# Learning Resources

## Online Sources – Digital Literacy

**Description**

- Digital literacy includes skills such as:
  - Asking relevant questions (i.e., on online forums)
  - Determining whether sources are relevant
  - Determining whether sources are trustworthy and high quality (NOTE: As discussed in the second quote, strategies for determining this for technical or coding strategies may differ than general information literacy. For example, if someone posting a technical solution has a lot of positive ratings, it may indicate that their code usually works well.)
- Online sources include:
  - Static websites, blogs, and wikis
  - Online communities and groups (e.g., sites such as Stackoverflow, blogs, forums, reddit, and interactive tools such as slack)
  - Webinars, podcasts, videos, and other multimedia content
- NOTE: Many sites and sources are now a combination of the above.

### Quotes

> Do your research online, I mean that's probably one of the most essential skills. We look for and also will lead to success, especially in a role like mine, because, especially in a start up there are limited resources. If there's something that I want to learn about. I'll just type it into Google and there are a couple of sites that I generally viewed to be trustworthy. For me as a web developer there's Mozilla network Doc and also Stack Overflow is very important. If it's more like a niche of a problem that I'm having or something I'm trying to learn, even just like the actual projects documentation, or like a forum for that project. And then also I do tutorials online. If I'm searching how to do this composite transform you know, maybe it comes up with a tutorial about composite transforms and then I'll just click on that and go through it. (Interview)
>
> A lot of times we're knowing what sources it's coming from and double check in the reliable sources and stuff. Iif you've seen it in more than one place and then you look at the comments on there, making sure you're reading the comments that people are saying. You're always going to get some negatives out there, but let's make sure the positives outweigh the negatives. Or a review online. … Someone's going to put a nice little review out there, but let's get someone that's actually done it and used it and had a good experience with it. It doesn't have to come from the vendor. There's a lot of other reliable sources like Reddit and you'll have peer type people and but you're going to know a lot of times from those pure relationship is this guy a contributor that's highly rated. And they have rating systems for us on the people that have given answers and stuff before to say hey, this guy knows what he's talking about. Making sure there is at least the bulletin board you run does have a rating system.. (Interview)

## Other Written Sources

### Description

A variety of other written sources are used, including the following. These may come from publicly available sources, sources only available to those within an organization, or proprietary sources provided by vendors or others.

- Books
- Documentation
- Research reports
- Specifications and standards
- Source code
- Trade press (journals/magazines aimed at members of a particular industry)
- Popular news sources

### Quote

> I spend a lot of time on reading trade press and trying to think through 'what does this mean.' You know, not just superficially what's going on, but what direction does this likely mean the industry is going to take? (Interview)
>
> My news feed on my phone is full of just like you know tech stuff (Interview)
>
> I'll look through sometimes if it's more like a niche of a problem that I'm having or something I'm trying to learn, you know even just like the actual projects documentation (Interview)
>
> They learn a lot 'cause they do it themselves. They don't have to read all the manuals and encode, you know, put all the pieces together… but they still learn a lot from that and they gain

> confidence from the beginning to the end to get this done. After you've done that, then you know you will expand your horizons and you will have to look at the documentation. But you know, you have a better familiarity with what's going on in this database platform. (Interview)

### Non-Formal Education

**Description**

Non-formal education refers to training that occurs outside of formal educational settings (including the traditional education system of primary, secondary, and tertiary education). In some organizations these are required. In other cases, computing professionals elect to engage in non-formal learning to update their skills (e.g., to obtain a certification, which will help find a future, more desirable job) or as part of a larger self-directed learning plan (e.g., to learn a specific new tool or language). Types of non-formal education experienced by participants include:

- In-person or online courses
- Boot camps
- Certifications
- Conferences

### Using Multiple Strategies and Resources

**Description**

Learning how to learn allows self-directed learners to be savvy users of learning strategies and resources.

Self-directed learners tend to combine multiple strategies and resources in order to meet their learning needs. For example, they may begin with a course to get an overview of a topic, then download sample code and engage in tinkering. If they come across an intractable error, they might look it up or ask a question on a forum, or they might decide to watch a youtube video to get more in-depth knowledge in one particular area.

## Mentorship

As described in the Lifelong Learning group, self-directed learning is common in this field. However, mentorship is also highly important in helping new employees or those new to a position or role succeed within an organization. Mentors help enculturate new team members. They also serve as an important resource for professional development. For example, a more senior developer might sit next to a mentee to help them with a coding problem – thereby modeling good coding practices as well as assisting with one particular problem, while a manager might mentor tech leads on how to go about being a better mentor for their own mentees.

**Description**

- Considering mentees' current skill level. Recognizing where they do not have skills or confidence. and assisting/mentoring and designing training programs
- Mentors can use a variety of techniques, such as:
    - Engaging in one-on-one or group discussions.
    - Providing advice. This requires understanding your team member's career goals as well as their upcoming tasks.
    - Providing feedback on technical tasks, writing, etc.
    - Modeling a skill or collaborating on a task that requires new skills, knowledge, or approaches.

    - Providing opportunities for growth and success.
- Managers in particular can provide mentees opportunities to develop skills that they will need in their current or future roles – where possible, help support their long-term career goals.

**Quotes**

> If it's a junior engineer I'll spend more time there than I will with a more experienced engineer. And then for my squad leads, it's really two-fold. I spend a lot of time with them in sessions where I help them work through project planning and implementation planning. So, a lot of architecting. And, our squad leads and team leads are also responsible for some people management side of things. So, we work through a lot of people issues as well and just kind of giving advice on how to handle different situations. (Interview, Manager)
>
> My junior engineers are [all] asking the same question. When is the best way for me to learn new things? And sometimes it's literally me asking them why do you want to learn? I will give you a project that forces you to be that self starter 'cause you have to finish something and so that means you have to learn it and. (Interview, Manager)

# Project Management for Individual Contributors

While managers and project managers require experience with a large number of management and project management skills (to be included in a separate document), computing professionals require some project management capabilities to stay organized, meet individual goals, and contribute appropriately to team work.

## Applying Project Management Models & Methodologies

- Includes traditional Software Engineering process models such as waterfall, spiral, rapid prototyping, etc as well as agile methodologies (e.g., Kanban, Scrum, Lean)
- While managers and project managers are responsible for the overall planning, every individual contributor must take part in managing the overall project and/or following the model that is set

**Quotes**

*It also indicates that the trend of software development cycle has been shifted from the traditional project management models to the agile project management models. This does not mean that the traditional software development models are not being utilized in industry. There are many companies that use a combination of traditional software development and agile process. Agile development is being specified most frequently compared to other agile project management processes such as Scrum, Lean, Kanban, etc. Traditional project management models such as the spiral or evolutionary models have not been specified by any of the job postings but only the waterfall model has been specified along with the agile process as a required background. These job descriptions required understanding of both agile and waterfall developments. (Literature – Kang et al., 2020)*

*We use agile software development life cycle. Normally we use a two-week Sprint. Sometimes when we have like a hackathon or holiday we do like a three week but normally 2 weeks and at the end of each Sprint we will have the in-team Sprint reveal. Every engineer and even a quality assurance engineer, they will demo what they have done in the past Sprint, so that would fall in the middle of the month. At the end of the month as a team our product team demos, we present what we have done in the past month to the entire company. (Interview)*

*The most popular agile software development method turned out to be Scrum, either used exclusively or together with Kanban. (Literature - Jarzebowicz & Sitko, 2020)*

*This skill is needed due to very dynamic changes in GSD Scrum projects while also considering time constraints. Many Scrum practices are used in the GSD context, such as sprint backlogs and sprint planning, daily Scrum,*
*and sprint review meetings. Thus, the development team must have Scrum expertise. (Literature - Hidayati et al., 2020)*

## Organized

**Quotes**

> I think that's mostly organizational and keeping up with... There are annual or static costs when they need to happen and then additional ones. Looking forward to replacement things that have to happen because there's always hardware and physical things that are rolling on and off, but a vast majority of that is just being organized and keeping up with that, usually through, I myself work better with just internal calendar invites, and because that's something I look at every single day. And lists and then whenever I do my budget. So I pull up last year's budget. I know which lines I need to pull over. (Interview)

## Goal Setting

**Quotes**

> Software engineers are capable of setting goals for themselves and working towards these goals.(Literature - Sedelmaier & Landes, 2014)

## Time Management

**Quotes**

> I think College in general teaches you to live on your own, and learn to fend for yourself, but the degree in general there is learning to work under deadlines learning to generate writing assignments on your own, especially as you get to the higher levels where an assignment is often a 14-page writing assignment. Now, I oftentimes have to read a 14 page very boring policy document or something like that. But knowing how to work under a deadline is something that is a learned skill. I feel so no matter what you're doing and what your degree is in, organization is required to keep you on that task. (Interview)

> Task Synchronization, which refers to the maintenance of sequential task accomplishment is reported by development team members as being an object of concern in order to not generate delays and/or blocking tasks. Thus, task inter-dependence increases the need for synchronization between them. Task Synchronization relates to both the activities between development team members (internal) and their activities with external agents (test team). "The delay in my task may delay a schedule because we have the scheduled start date for testing. Because of this, we need to keep our activities in sync. We are dependent." (Literature - Marsicano et al., 2018)

# Strategic Processes

This set of competencies goes to the core of the mental work that goes into designing an effective solution to address the type of ill-structured problems computing professionals face. While there are dependencies between all of these competencies, *Problem Solving* and *Decision Making & Making Judgements* are tightly intertwined with one another as part of the design process computing professionals go through as they assess the situation, gain understanding of a problem, and work towards solutions. The *Thinking and Comprehension* competencies, including Critical Thinking, Critical Reading, and the ability to Think on your Feet are foundational skills, especially when dealing with ill-structured problems.

These are among the more difficult to teach competencies. In fact, even if an individual is a strong critical thinker or problem solver in one domain, their ability may not transfer easily to another domain [CITE]. Therefore, continued engagement in opportunities to develop these skills is crucial for developing a confident and competent computing graduate.

## Empirical Research Skill

### Description

- Collecting and analyzing and/or reading and synthesizing research data
- While not mentioned as frequently as other competencies, this is an important part of the job for some computing professionals.

### Quotes

> Research skills (Literature – Schirf & Serapiglia, 2017; Berkling et al., 2019)
>
> Research methods and empirical validation (Literature – Mikalef & Krogstie, 2019)
>
> Audiovisual Understanding - to collect and evaluate contents visually and/or auditorily (Literature – Gold & Sedelmaier, 2014)

## Decision Making & Making Judgements

### Description

This competency relates to factors that go into decision making. Although there are tools and techniques to balancing these factors, experts use *design judgement* to guide their decision making. There is generally no "right" decision or solution – rather, a solution needs to be selected and designed/developed based on multiple factors in a way that will vary between situations and contexts. The sub-competencies in this section relate to some of those factors.

Participants stressed that this is very important to consider during educational programs.

### Quote

> I think we get so focused on teaching, 'here's how you program in C,' 'here's how you program in Java,' 'here's how you program … data structures.' We forget that we aren't teaching them when some of those things are appropriate, and when sometimes it's better to step back and say, hey, how about simpler?' (Interview)

## Determining Acceptable Risk

### Description

- Assessing risk associated with potential failures
- Practicing caution and increased attention to detail when risk is higher

**Quotes**

> If there is something …that I look at it and I determined myself that it's fine, and then come to find out it is actually something that somebody used to compromise the company, that's on me. We don't always have to be right but we always need to double check everything we do. You do need to be very, very careful in this workspace because there is a very large margin for error. And the consequences for that error can be very bad. All the way up to a company going out of business. (Interview)

## Assessing Business Aspects

**Description**

- Assessing business implications of a decision. This may require knowledge of the market; business acumen, ability to do a financial analysis and other business-related skills; and/or the ability to communicate technical issues such that another individual with that skillset can make a well-informed decision.
- Participants mentioned that in software engineering, the later you make a change, the more costly it is. Some ways to address this fact include spending more time in the design stage or using Agile or similar approaches.

While many of the quotes were by mangers, computing professionals early in their careers may be asked to provide information that may impact decisions, such as getting quotes for different software packages, vender services, etc.

**Quotes**

> Actually sitting down and doing the financial analysis, let's crunch some numbers here and they don't need to be stupendously precise. They need to be order of magnitude numbers. Recognize having that that those skills of kind of being able to put a finger in the air do a few quick calcs and say, yeah, we're in the right ballpark here. (Interview, Manager)
>
> I talked to the CEO and we're like, 'well, why are we doing this?' And after we talked [it] out for about 30 minutes, it became clear that we didn't. It wasn't a business driver anymore. We had other sources of business drivers that were more important. (Interview, Manager)
>
> It's very costly to change your design or anything later in the product lifecycle. Again, there are different models to it that people say, let's design it. And then when we finished the design and go to implementation phase then we never looked back at the design. No, It's an iterative process as you learn some new information, new things, then it's going to change in the early [stages, so that you are] not wasting a lot of time. If you changed later it, it gets costly. Think about, in real life you bought a car and then suddenly the car company says, oh, we have this particular issue and you have to bring the car back to the dealership. How much does it cost for the car company? It's billions. But if they had done it right in the 1st place, you know got that issue very early on while the car was in production or preproduction, whatever is before they even shipped it. They would have saved like millions of dollars. (Interview)

## Balancing Cost, Time and Workload

**Description**

- When making judgements, considering the balance between cost vs time to complete vs workload for team members. These judgements may vary depending on the situation and context.
- While this might happen at a managerial level, similar decisions have to be made at an individual level, especially in determining when to proceed down one path vs trying another approach, or recommending a different solution

**Quotes**

> I had to learn it all and there were a few cases where I started down a path and realized I don't know how to do this and it's going to take me too long to learn it. I don't have the time, so I need to find a different way to do it that's easier and requires less new knowledge on my part. Another little corner of the skills is understanding when you're going down a path that's not going to lead to success and realizing that you're doing that and getting off that path and finding another way quickly. (Interview)

> It is part of [some people's] core personality to overdesign because, they enjoy that. Sometimes it's completely appropriate. And sometimes it's even underdesign. There's sort of two things that you're balancing. And if you understand the problem well enough in your designing towards that as an optimal, then you may not be over designing. But if you're spending a lot of time and you're driving down the road of optimization towards a solution and you misunderstand the problem, that's hugely wasted effort. (Interview)

## Considering Customers' Needs

**Description**

- Taking customer's and users' technical and business needs into account in requirements, design, and other decisions

**Quotes**

> Learn what various customers needed or wanted and ensuring that the network was provisioned in a way that they could achieve what their goals were. Researchers [end -users] often times needed to do things that were not the norm and so we need to make sure that we had ways to adapt for that. Understanding the whole as more than the sum of the parts, understanding how the pieces fit together and understanding more about what your customers are doing on your service so that you can understand better what impact you have when something goes wrong or needs to be adjusted. Yeah, I think we can't be siloed anymore. (Interview)

## Knowing When to Escalate or Ask for Assistance

**Description**

- Knowing when to reach raise priority or call in others to assist

**Quotes**

> When do we escalate, because that's another thing I see is people escalate too soon or without understanding. So after a while it's like crying wolf. So people lose respect for you when you actually have a real issue. (Interview).
>
> So I think the most important thing for me was just knowing when to reach out to help, not procrastinating (Interview)

## Considering Others' Competencies

### Description

- Recognizing competencies and capabilities of customers, vendors, employees, and other stakeholders
- Determining what they know and can do and not pushing them beyond what they are able to do
- Determining when they are struggling for the purpose of accommodating or providing guidance

### Quotes

> Vendors say they're going to do 'blah blah blah', and if you think about where is the vendor positioned in this industry and would this really be to their advantage to do it? Do they actually have the techniques? Have they demonstrated the technical capability to do something that they're at the of the scope of what they're talking about doing? (Interview)
>
> Something just goes wrong and OK, the skill gap is there, and we found out the hard way because we didn't ask the right questions or we didn't dig into stuff and as a result, we ended up with a system that the predictions weren't as accurate as we thought they were going to be. The data scientists and the software engineers were not translating requirements well enough. And so, what actually got implemented was not what got prototyped. But now we're like discussing it and coming up with, OK, how are we going to fix this going forward. (Interview, Manager)

## Assessing Technical Aspects

### Description

- Assessing whether a potential solution is technically feasible
- Assessing technical impacts of a decision
- Weighing different technical solutions based on their merits and other factors. This may, for example, lead to recognizing when a less than ideal technical solution is acceptable and a more rigorous, efficient, or maintainable solution is not worth the time, cost, or workload that would be required.

### Quotes

> The perfect is the enemy of [the] good. So, implementing something is better than nothing, but it's finding the balance between 'this will work for five years' or 'this[will work] for 50.' ….
> The banking software was written 50 years ago, [or] that [is] used right now probably won't exist in 100 years. But all the transactions that it executed, all the people that it helped, it's at least valuable during that time span. It's the epoch of its usefulness. (Interview)

> These are the problems that have mapped them to what algorithms should I apply. So thinking about it more. How do I apply the code? Now it's come to a point where you have this abundance of choices. It's like what library do I use in this circumstance? Or if I was like looking at it from a data science standpoint, it's OK, this algorithm is good for this type of data. If I wanted to do like image recognition, I would want to use these certain things and I think that's what I struggled with when I was learning. (Interview)

# Problem Solving

## Collaborative Problem Solving

**Description**

- Ideating with others and building on one another's ideas
- Engaging in collaborative dialogue towards solving a problem

Note that this is something that might be assessed in a technical interview, as described in a quote below.

**Quotes**

> [During an interview] we bring them in for what we call a 'white boarding exercise'. We basically get [another engineer] in a room… we present a problem to them and we asked them to architect a solution with us. So not like, 'oh you get up at the white board and you solve it.' But 'let's solve this problem together,' and we see how they think through problems, how the back and forth goes with the discussion and the idea, you know, bantering basically, and what it would be like to actually architect a solution with this person. (Interview, Manager)

> Sometimes we find that… there's a very real security issue and we work with the security team and IT [team] to assess how can we best defend against this. Does that mean we need to quickly apply some patches? Do we need to provide some access controls to limit access to whatever is at risk? Maybe the security issue only is accessible if three conditions lineup. Is there some way we can take a step to make sure that one of those conditions no longer apply? (Interview)

## Ill-structured Problem Solving

**Description**

- Dealing with complex problems that have no one "right" solution. Sometimes called "wicked problem solving," because these problems are not easy to tackle.

**Quotes**

> This is my absolute favorite, mostly because there's no fix yet that's been published, so I have to figure out a way to mitigate the risk without really having any feedback from the rest of the field. So it's that whole not knowing, but knowing that there's a problem in that, the drive to fix the problem as fast as possible. And to make sure that when I get asked by our customers or our vendors what steps we took, I have great pleasure of telling them that we immediately started addressing it even before the official fixes are available from the vendor. (Interview)

## Problem Solving Approach

## Understanding the Problem

### Analyzing & Identifying the Problem

**Description**

- Developing a complete understanding of the problem and understanding all specifications
- Analyzing problems and identifying challenges and restrictions
- Identifying and formulating practical problems that theory does not explain

**Quotes**

> I think the first part is extracting the problem, understanding it getting more information, and doing more research. That's the first phase and once you understand the problem, maybe even run that by a few people just to make sure that you can explain the problem. Because if you can't explain it, you probably don't understand it. There's sort of two things that you're balancing, do you understand the problem well enough? And if you understand the problem well enough are your designing towards that as an optimal, that you may not be over designing. (Interview)

### Defining the Problem

**Description**

- Problem framing
- Determining exactly what you are trying to solve
- Determining the scope

**Quotes**

> I think that the first thing to do is get your hands around the problem, right. You have to define the problem and lots of times I've found that I'm tempted to jump into solving the problem without fully defining it, and that rarely ends well. So first thing to do to solve it is to get your hands on it, hands around it, and then you have to not just get your own hands around it, but you have to get the stakeholders in your organization around it and that could be even people outside your organization. And sometimes that actually redefines the problem, and sometimes the problem goes away. (Interview)

> That's definitely one thing we want to make sure. We document clearly what are the pain points that we are trying to solve? Make sure everybody signs off and then we start working on. (Interview)

### Pattern Recognition

**Description**

- Recognizing types of problems and the types of solutions/designs that may be appropriate

**Quotes**

> Different architecture approaches should be adopted under different circumstances (Literature – Ho & Framptom, 2010)

### Understanding the Bigger Picture

#### Description

- Understanding how the piece they are working on fits within the larger software application

#### Quotes

> Look at a different solution to the problem. Look at it in a way that I may not have thought of before. Kind of knowing how the back end of that works. Not just knowing that something works, but knowing how it works is useful because you might be able to look at the whole picture and come up with either something that's simple, or something that works a bit better. What I do is looking at that pattern. You can look at one thing and that one thing is fine. But what you really need to do is look at the whole picture. What something does like a programming language, you code and it outputs what you want, but it's also good to know the overall idea of it. It helps me problem solve a bit faster and a bit, I think more effectively. (Interview)

## Brainstorming

### Abstraction and Conceptualization

#### Description

- Thinking abstractly about the problem and potential solutions
- Mentally visualizing potential solutions

#### Quote

> These skills are to invent or contrive an idea and formulate the idea mentally. All interviewees mentioned these skills in the interview. One interviewee explained, 'it is the ability to visualize complex structures in their mind.' Other interviewees identified - capacity to imagine, visualize a solution, the ability to conceptualize things or concept -building, seeing concepts, working with concepts. Those imagining skills that allow you to think in terms of models or concepts and allow you to form relationships between those concepts and even think about and build abstract machines in your mind's eye. (Literature – Ho & Frampton, 2010)

### Breaking Down a Problem

#### Description

- Breaking a large, complex problems into its composite parts/sub-problems
- Helps to better understand the problem
- Helps to identify potential solutions of the sub-problems identified

#### Quotes

> But really breaking it down, breaking up these things into concepts, whether that manifests itself as functions or individual programs or API endpoints or something like that, depending on what you need to get done. Being able to take this task may get three smaller tasks and then figure out the best way to make these three tests do what they do.

> Learning how to breakdown a problem. We do that with programming. Sometimes we break it down, but we don't know why we're doing it... Teaching [students] when it's appropriate to pull something out into a subroutine as opposed to not doing that. (Interview)
>
> We take it in and try and break it down to figure out how widespread that problem is to see where we can then start figuring out the problem. Is it a security problem? Is it networking? Is it storage? (Interview)

### Ideation

**Description**

- Brainstorming to come up with multiple ideas
- Noting all possible solutions/ideas even if they seem bad
- Fleshing out solutions to understand the implications

**Quotes**

> There needs to be brainstorming, there needs to be an idea, coming up with different ideas and what are potential solutions. Even if something seems bad it is just good to write out all the different ways that you could potentially solve this problem. I think, to just kind of flesh out each of those solutions, just to get a good idea of either what it would look like, what it would mean to the application. (Interview)

### Looking for Existing Solutions to Similar Problems

**Description**

- Researching if something similar has been done by someone else before (could be within or outside of the organization)
- Applying a similar approach or taking inspiration from existing solutions *(this might also be called 'precedent use' in design fields)*

**Quotes**

> So, I think that the lowest hanging fruit, is this a problem that has already been solved either by me or somebody else? So, I mean, if it already has some, then go for that as opposed to trying to reinvent the wheel. I mean, reinventing the wheel has certainly been a problem in this organization. (Interview)

## Strategizing Solutions

### Considering Multiple Solutions

**Description**

- Considering multiple solutions before selecting one
- Realizing that the first solution isn't always the correct solution and determining when to move on to trying other solutions
- Not fixating on one technology, tool, or language
- Using analysis and judgement to determine which to select *(See Decision Making & Making Judgements)*

### Quotes

> Once you kind of get that down, then you kind of need to narrow it down to a decision because it's great to have a lot of different ideas, but in the end you actually have to try and solve the problem, and you have to come up with a first way to do that. What I generally do is, a list of pros and cons for each of the solution. I try and look at what are the priorities, what may be a problem, usually problems are broken down into many sub problems. And then you need to say what problems are easy to fix, what can I do right away what's not going to have an impact. For all these different solutions, you have to analyze each of them and determine is it really in line with the priorities like what are the pros and cons and then you just have to settle on something that you start with. (Interview)

> The ability to identify various alternatives, alternative approaches and analyze the implications of going down one path against another. Eight interviewees emphasized that being able to consider alternative solutions was important - the role demands an ability to identify the potential solutions and when you set your mind to a problem in the abstract, you can often come up with multiple ways of solving it, and, generally speaking, you can look at those alternatives and you can work out pros and cons to all of them. Different architecture approaches should be adopted under different circumstances. (Literature - Ho & Frampton, 2010)

> I think that sort of process of taking the idea of putting it down, iterating that, getting it to a point, and you may throw it away. I've been in several discussions where we go into a meeting and we said OK, that's good. We're going to set that aside because now the conversation is evolved to a completely different thing. And I initially had thought that was wasted effort, but it really wasn't because you thought about all the different avenues. I guess the other skill is, something that happens in computing a lot, not relying on specific knowledge of specific technologies as a crutch for problem solving… It would be a good soft skill to not be so reliant or be so transfixed on a particular language. (Interview)

## Prototyping, Piloting and Modeling

### Description

- Creating a prototype to share with others for feedback
- Testing or "experimenting with" initial solutions to ensure they address the problem and do not create new problems

### Quotes

> You create an environment where I can actually experiment based on a mockup of the idea. I can actually come up with a set of ideas, broad ideas about my work, and I can actually deploy them in some easy way. Do I change the size? Do I change the font? There are so many things you could do. Several participants took it upon themselves both to design incentive systems that get users to adopt a product feature and to create user telemetry and surveys that measure whether the systems worked. So we create a game that gets people to repetitively use the feature. And then we watch what happens when we take the game away. Did it stick or did it not stick? (Literature – Kim et al., 2016)

> At least try to fix it in a in a sandbox or testing environment, and then once that's of course tested and verified you know that it doesn't break anything else. Then it could be rolled out to production once it goes through the form of the change management process. (Interview)

### Reflection-in-Action

**Description**

- Reflecting throughout the design process, which allows one to modify design strategies as they continue to design
- This may include conducting 'experiments' which serve to generate both a new understanding of the problem and of the potential solutions

Note: As explained in Schön (1987), this is in contrast to reflection-on-action, which occurs after the design process has occurred.

**Quotes**

> Like journaling. You know, it's a way to sort of write your thoughts out. You're not writing a journal necessarily. For documentation purposes, you're writing it as a way to slow your thinking down to put it in a different cadence so that you can think about it a little bit differently. Because your mind is racing and you don't have to place things in a physical space, but when you write them down, then you have to place them there and you go back and read the words and they may change in meaning. It's a wonderful thing about languages that you can read the same thing two times in being different places in your life. And it means different things. (Interview)

### Iterating on a Solution

**Description**

- Refining a potential solution
- This is typically a continuous or multiple-step process

**Quotes**

> Can you build a better version of the product? …No one is using it and if we build this way then you know people will use it so it's a very iterative process and no one wants you to solve in a day or two. You know it's a long process, is an iterative process. You just keep on working on it and improvise or there. (Interview)

## Implementation

### Communicating the Rationale for Solution

**Description**

- Clearly articulate and justify your solution

**Quotes**

> [When hiring, I look for] the ability to explain what you've done because you're not any good to me if you can solve the problem, but you're the only one that understands what you did because somebody else is going to have to come behind you. So, there's some communication that's needed as well there. We get recordings of how they solved the problem, so we also get their thought processes. How they approach the problem as they are solving it. (Interview, Manager)

### Deploying Solutions in Phases

**Description**

- Prioritizing parts of a solution and deploying as planned.

**Quotes**

> Sometimes you just have to start working in small batches and then see where it goes because it's a lot of times it's hard to get the whole big picture. One thing is working with the client at different steps. If there are 10 steps to go from the start to finish, you don't wait for all the 10 steps to be done before you show what you have, but show it every step of the way. So because people don't really know what they want until they actually see it. And there's a lot of tweaks. It's easier to do it in the beginning, rather than towards the end. (Interview)

### Using Technical Skills and Knowledge to Implement Solution

**Description**

- Relying on technical competencies to implement a solution

**Quotes**

> *[Differentiating what a software developer vs a data scientist would bring to the problem, based on their prior knowledge:]* As a software developer] I know if you're using a hash map in a programming language like, you're trading off space for speed like you're just going to gobble up memory, but you're willing to do that because you're going to get order one look ups and that's fantastic. And you also don't want to be reallocating these things because it takes time to do that. So you're gonna like reserve a bunch of space at the beginning end. Like, I wouldn't expect the data scientist to know those things. And so …when you make a decision to use a certain solution, there's a bunch of kind of like implicit things that an expert would know that somebody outside wouldn't know that. Oh, when you're using this type of data science model, you have to look at the results when it's not working, it's just gonna spit out zero. (Interview)

## Thinking and Comprehension

### Critical and Analytical Thinking

**Description**

- Use logic to analyze a situation and make reasoned judgments
- Being able to look at something and figure out what is required to determine whether or not a task is completed successfully

**Quotes**

*One of the things I think is missing from a lot of our technical education is the why. We learn the how, we learn the what, but we don't always learn the why we do something. And I think, while it's harder to teach that it takes a lot more time. Because I believe if you know why we do something, then when you come across something that's a little different you could now start making some educated guesses about how should I approach this based on why we went another way. What does that tell me? And so I think that's something that's missing from a lot of education is why. (Interview)*

## Critical Reading

### Description

- Engaging in deeper examination of text while reading for enhanced clarity and comprehension
- Reading techniques and information processing to understand aids and literature correctly
  e.g. data sheets

### Quotes

*Being able to read documentation is also very huge because there's a bunch of different things that we are trying to integrate. So being able to parse through that relatively quickly and not take weeks basically to understand an API is very important just to keep things moving smoothly.*

## Thinking on your Feet

### Description

- Making decisions quickly
- Seeing through the implications of making a decision on the spot

### Quotes

> Sometimes you are given a problem or you encounter a problem which you have never seen before right in that particular case, what you're going to do. Still here want to put your thinking hat and try to solve. Try to think on your feet. And you don't have much time. You have like half an hour and sometimes an hour or two, and you're given set of problems here. (Interview)
>
> Being able to kind of think on your feet when you're confronted with technologies that you may not fully understand. (Interview)

# References


Berkling, M. A. Incekara, & T. Wolske. (2019). A Study of Dispositions According to the IEEE Information Technology Curricula 2017 for German Industry and Student Population. *2019 IEEE Global Engineering Education Conference (EDUCON)*, 237–244. https://doi.org/10.1109/EDUCON.2019.8725211

Chugh, M., Chanderwal, N., Upadhyay, R. K., & Punia, D. K. (2021). Antecedents and consequences of knowledge sharing for software process improvement in the Indian software industry. *Journal of Software: Evolution and Process*, *33*(2).

Cummings, J., & Janicki, T. N. (2020). What Skills Do Students Need? A Multi-Year Study of IT/IS Knowledge and Skills in Demand by Employers. *Journal of Information Systems Education*, *31*(3), 208–217. ERIC.

Hidayati, A., Budiardjo, E. K., & Purwandari, B. (2020). Hard and soft skills for scrum global software development teams. ACM International Conference Proceeding Series, 110–114. https://doi.org/10.1145/3378936.3378966

Ho, S. Y., & Frampton, K. (2010). A competency model for the information technology workforce: Implications for training and selection. Communications of the Association for Information Systems, 27(1), 63–80. https://doi.org/10.17705/1cais.02705

Janicki, T. N., Cummings, J., & Kline, D. (2014). Needs and Implications for Information Technology Course Content. Information Systems Education Journal, 12(6), 59–70. ERIC. https://files.eric.ed.gov/fulltext/EJ1140765.pdf

Jarzebowicz, A., & Sitko, N. (2020). Agile requirements prioritization in practice: Results of an industrial survey. In J. L. C. Cristani M. Toro C. ,. Zanni-Merk C. ,. Howlett R. J. ,. Jain L. C. (Ed.), Procedia Computer Science (Vol. 176, pp. 3446–3455). Elsevier B.V. https://doi.org/10.1016/j.procs.2020.09.052

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic Inquiry*. Newbury Park, CA: Sage.

Kim, M., Zimmermann, T., DeLine, R., & Begel, A. (2016). The Emerging Role of Data Scientists on Software Development Teams. Proceedings of the 38th International Conference on Software Engineering, 96–107. https://doi.org/10.1145/2884781.2884783

Marsicano, G., de Oliveira, V. L., de S. Mariz, L. M. R., & da Silva, F. Q. B. (2018). An Initial Understanding of Task Interdependence in Software Engineering: A Case Study. Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering, 21–28. https://doi.org/10.1145/3195836.3195851

Mikalef & J. Krogstie. (2019). Investigating the Data Science Skill Gap: An Empirical Analysis. 2019 IEEE Global Engineering Education Conference (EDUCON), 1275–1284. https://doi.org/10.1109/EDUCON.2019.8725066

Ruff, S., & Carter, M. (2015). Characterizing Employers' Expectations of the Communication Abilities of New Engineering Graduates. Journal on Excellence in College Teaching, 26(4), 125–147. ERIC.

Schirf, E., & Serapiglia, A. (2017). Identifying the Real Technology Skills Gap: A Qualitative Look across Disciplines. Information Systems Education Journal, 15(6), 72–82. ERIC.

Schön, D. A. (1987). *Educating the reflective practitioner : toward a new design for teaching and learning in the professions* (1st ed.). Jossey-Bass.

Sedelmaier & D. Landes. (2014). Software engineering body of skills (SWEBOS). 2014 IEEE Global Engineering Education Conference (EDUCON), 395–401. https://doi.org/10.1109/EDUCON.2014.6826125