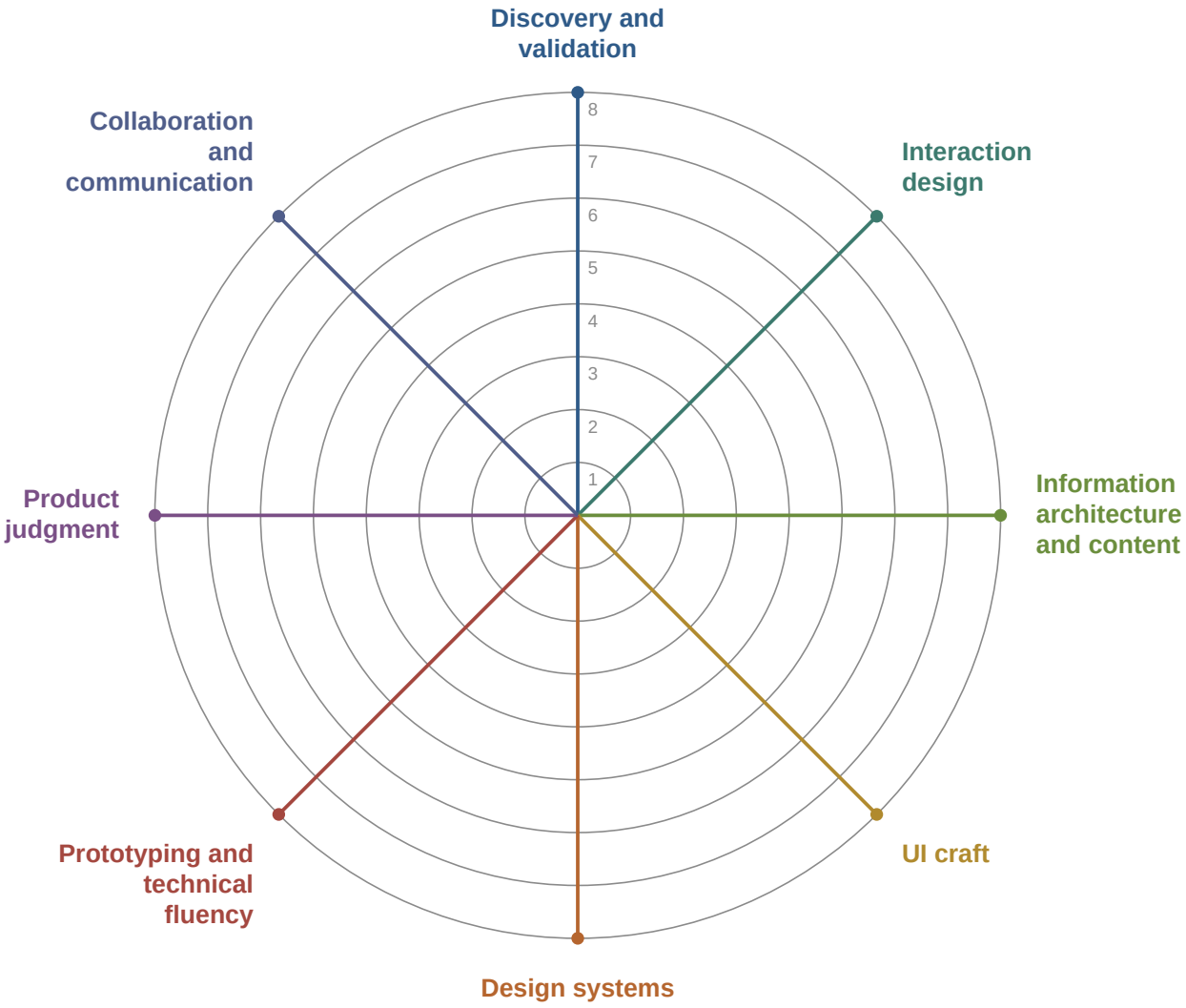
Product Designer Skills Wheel

Eight skills for assessing product and interaction designers, modeled on Petra Wille's PM Wheel. Each skill has a description and a set of questions to help the designer and their manager understand what the skill covers and where the designer stands.

Score each skill by marking a point on its line, 0 at the center to 8 at the edge, then connect the points. Use the questions on the following pages to decide where you stand. To compare, mark the self-assessment and the manager's assessment in different colors.



Name: _____ Assessed by: _____ Date: _____

1. Discovery and validation

Framing the problem, then testing ideas with users through usability tests, quick interviews, and analytics. Research at the depth a product designer practices it, not a full research program.

1. Before starting design work, can you state the problem in a sentence or two without describing a solution?
2. Do you know who the users are for what you're designing, and how their needs differ?
3. Do you look at what's already known (support tickets, analytics, past research, sales calls) before asking for new research?
4. Can you tell when a problem needs validation and when it doesn't?
5. Do you write down your assumptions and identify which ones are riskiest?
6. Can you plan and run a usability test on your own?
7. Do you write test tasks that don't lead the participant?
8. Can you run a user interview without pitching or defending your design?
9. Do you talk to users regularly, not only when a project calls for it?
10. Can you read product analytics well enough to find where users struggle?
11. Can you recruit the right participants, including hard-to-reach ones?
12. Can you separate what users say from what they do?
13. Do you synthesize findings into a few clear conclusions rather than a list of observations?
14. Do your findings change your designs, and can you point to examples?
15. Do you share what you learn with product and engineering in a form they use?
16. Do you check after launch whether the design did what it was supposed to?
17. Do you know when you've learned enough to move forward?

Score: _____ Notes:

2. Interaction design

Flows, states, edge cases, and error recovery. Thinking past the happy path.

1. Do you design complete flows rather than individual screens?
2. Do you design empty, loading, error, partial, and success states without being reminded?
3. Do you find edge cases before engineering does?
4. Do your error messages and recovery paths tell users what happened and what to do next?
5. Do you account for permissions, account types, and other conditions that change what a user can see or do?
6. Do you design for both first-time users and experienced users who repeat the task daily?
7. Do you reduce steps and decisions without hiding necessary control?
8. Do you use patterns users already know before inventing new ones?
9. Can you explain why a given interaction pattern fits the situation?
10. Do you design for keyboard and screen reader use as part of the interaction, not afterward?
11. Do you consider what happens when actions are slow, fail, or partially succeed?
12. Do you design destructive actions carefully, with confirmation, undo, and clear consequences?
13. Do your flows handle long-running and asynchronous work, such as uploads, imports, or processing?
14. Do you treat every surface customers touch (email, notifications, help content, APIs, other channels) as part of the product, and design flows that move across them?
15. Can you diagram a complex flow so others can follow and critique it?
16. Do you design responsive behavior where it matters?
17. Do you spot interaction problems in the shipped product and propose fixes unprompted?
18. Can you simplify a complex workflow without losing capability?

Score: _____ Notes:

3. Information architecture and content

Structure, navigation, naming, and microcopy. Users can't use what they can't find or understand.

1. Can you organize a large set of features or resources into a structure users can navigate?
2. Does your structure match how users think about the domain, rather than the org chart or the database?
3. Can you run or interpret a card sort or tree test?
4. Do you name things consistently across every surface customers touch?
5. Do you push back when a name is confusing, internal, or inconsistent?
6. Do you write the UI copy yourself rather than leaving placeholder text for someone else?
7. Is your microcopy short, specific, and in plain language?
8. Do your labels describe what something does rather than what it's called internally?
9. Do you write error messages, empty states, and help text as carefully as primary screens?
10. Does your navigation scale as features are added?
11. Do you design for findability: search, filtering, sorting, and cross-linking?
12. Can you design pages that hold dense or complex information without overwhelming the reader?
13. Do you work with marketing, support, and documentation so every surface describes things the same way?
14. Do you maintain or contribute to a terminology list or content guidelines?
15. Can you explain the information architecture of the product you work on, including where it breaks down?
16. Do you consider how content reads when values are long, missing, or unexpected?
17. Do you test whether users understand your labels and copy?

Score: _____ Notes:

4. UI craft

Hierarchy, layout, density, polish, and accessibility. Making screens clear and finished, not merely correct.

1. Does your visual hierarchy make the most important thing on the screen obvious?
2. Do you use layout, spacing, and alignment deliberately and consistently?
3. Do dense screens, such as tables, dashboards, and settings, stay readable in your designs?
4. Do you understand typography well enough to make text easy to scan?
5. Do you use color for meaning without relying on color alone?
6. Do your designs meet accessibility standards for contrast, sizing, and focus states?
7. Do you design with real data: long names, large numbers, missing values, many rows?
8. Is your work polished at handoff, or does it need another pass?
9. Do you notice small inconsistencies in spacing, alignment, or style, in your work and others'?
10. Do you design dark mode, responsive sizes, and other variants when they apply?
11. Can you make a screen feel simple without removing what users need?
12. Do you review the built product and catch visual defects before release?
13. Can you explain your visual decisions in terms of the user's task rather than taste?
14. Do you know when a screen is done and more polish isn't worth the time?
15. Do you stay current on UI conventions for the platforms you design for?
16. Can you critique visual work specifically and constructively?
17. Does your work look like it belongs to the same product as everyone else's?

Score: _____ Notes:

5. Design systems

Thinking in components, keeping things consistent across surfaces, and extending the system without forking it.

1. Do you use existing components before designing new ones?
2. Can you tell whether a new need is a variation of an existing component or something genuinely new?
3. When you need something new, do you design it as a reusable component rather than a one-off?
4. Do you contribute components back to the system with documentation and usage guidance?
5. Do you understand how the system's components are built in code?
6. Do you use design tokens and system styles rather than custom values?
7. Do you keep design files organized so others can find and reuse your work?
8. Do you document component behavior, including states, variants, and when not to use it?
9. Do you spot inconsistencies across products and propose fixes?
10. Can you make a case for changing the system when it no longer fits a need?
11. Do you work with engineering to keep design and code components in sync?
12. Do you build accessibility into the component rather than fixing it on each screen?
13. Do you consider how a change to a shared component affects every place it's used?
14. When you break from the system, is it justified, and can you explain why?
15. Do you help other designers use the system correctly?
16. Do you know the system well enough to design quickly with it?

Score: _____ Notes:

6. Prototyping and technical fluency

Building testable prototypes at the right fidelity, including code or AI-built ones, and knowing what's cheap or expensive to build.

1. Do you match prototype fidelity to the question you're trying to answer?
2. Can you build a clickable prototype fast enough to test it the same week?
3. Can you build prototypes in code, or with AI tools that produce working code?
4. Do you prototype what static screens can't show, such as transitions, timing, and real data?
5. Do you use prototypes to answer questions, not just to present finished designs?
6. Do you understand the basic architecture of the product you design for?
7. Do you understand the data model behind what you design, and what the system can and can't do?
8. Do you understand the product's core concepts and how they relate, well enough to design without an engineer explaining each one?
9. Do you know which parts of a design are cheap to build and which are expensive?
10. Do you ask engineering about constraints early rather than discovering them at handoff?
11. Do you understand what data is available and how quickly it can be loaded?
12. Do you use the product the way customers do, on their devices and in their conditions?
13. Can you inspect the built product with the platform's tools, such as browser developer tools?
14. Do your handoffs specify the behavior engineers need without a meeting to explain them?
15. Do you understand how AI features behave (latency, variability, failure) well enough to design for them?
16. When you learn a constraint, do you adapt the design rather than insisting on the original?
17. Do you keep learning about the technology your product is built on?

Score: _____ Notes:

7. Product judgment

Tradeoffs, scoping to the smallest useful version, and tying design to user and business outcomes. This separates a product designer from someone executing a spec.

1. Can you explain the business goal behind the project you're working on?
2. Do you know how the work will be measured, and do you design toward that measure?
3. Can you scope a design to the smallest version that solves the problem?
4. Do you propose phased approaches when the full design is too expensive?
5. Can you tell which parts of a design matter most and which can wait?
6. Do you make tradeoffs explicitly and explain what you gave up?
7. Do you understand who the customers are, who pays, and how their needs differ?
8. Do you know competitors' products well enough to learn from them without copying them?
9. Do you consider pricing and business model where they affect the experience?
10. Do you push back on requests that solve the wrong problem?
11. Do you bring ideas for what to build, not only for how to design what you're given?
12. Do you consider the cost to support, sales, and operations, not only to users?
13. Do you know when good enough is good enough?
14. Do you follow up after launch to learn whether the design worked?
15. Can you argue for design investment in terms product and business leaders care about?
16. Do you understand the roadmap and how your work fits into it?
17. Do you make sound decisions when the data is incomplete?

Score: _____ Notes:

8. Collaboration and communication

Partnering with product and engineering from the start through ship, defending decisions with evidence, and giving and taking critique.

1. Do you involve product and engineering before your design is finished?
2. Do you share work early, while it's rough, rather than waiting until it's polished?
3. Do you take part in planning and estimation conversations?
4. Do you stay involved through build, answering questions and reviewing the implementation?
5. Do you resolve disagreements with product and engineering directly and constructively?
6. Can you explain the reasoning behind a design decision with evidence?
7. Do you change your mind when given a better argument?
8. Do you hold your position when you have evidence and the stakes matter?
9. Can you present work to executives in a way that gets a decision?
10. Can you present the same work to engineers with the detail they need?
11. Do you give critique that is specific, kind, and useful?
12. Do you seek critique of your own work and act on it?
13. Do you write clearly in specs, tickets, and Slack?
14. Are your decisions and the reasons for them easy to find later?
15. Do you keep stakeholders informed of progress and changes without being asked?
16. Do you help other designers improve?
17. Do you work well with people outside product and engineering, such as support, sales, and docs?
18. Do people seek you out for design input?

Score: _____ Notes: