problem-solving-101

Table of Contents

Introduction

In many situations, starting from interviews, onboarding, working in groups, and individually, with larger team constellations and so on it is very important to have a clear and understandable problem solving process, that is being followed consistently, and when side-steps are done they are done in an aware manner.

In all the other repositories, we are trying to have a proper problem solving approach, documented in the README.MD for each repository. We are not very good at documenting side-steps from it and sometimes time does not allow for it at all.

This expanded problem-solving process provides a more comprehensive view, starting from why the problem is worth solving, through how to solve it, and finally, how to validate the solution.

Lets have the tournament winner problem as an example.

Step 1 - Need identification (Why)

This is the stage where you identify the need for solving a problem. You consider the impact of the problem, who it affects, and why it's worth solving. For instance, in the tournament problem, the need could be "There is a requirement for a fair and efficient system to determine the winner in a series of games or competitions."

> **Team note:** At this stage, gather requirements from stakeholders. These could be product managers, customers, business analysts, or other relevant parties. Understand why the problem is important to them, what their needs are, and how solving the problem would add value. This stage may involve meetings, surveys, or user research to better understand the needs and constraints.

Step 2 - Problem Definition (What)

Here, you would articulate the specific problem you're trying to solve based on the identified need. In the tournament problem, this step would involve stating that you need to find the team with the most points, with each win granting three points.

> **Team note:** Based on the needs and requirements gathered, define the problem. Make sure the problem statement aligns with the stakeholders' understanding.

Step 3 - Conceptual Understanding (When)

It's important to understand the specific circumstances or context in which the problem exists. This could include understanding when the problem occurs, under what conditions, and any constraints that exist. In the tournament problem, the context could be "The solution should work for any number of teams and games, and should handle ties appropriately."

> **Team note:** Understand the context in which the problem exists. This could involve understanding the user journey, existing system architecture, business rules, and regulations, etc. Also, consider the design or user interface (UI) elements that might affect the solution.

Step 4 - Conceptual Solution (How)

This is where you devise a conceptual or theoretical solution to the problem. It might not be the final solution, but it forms a basis to start the actual problem-solving process. Here a simple example of pseudocode could help to make things clearer earlier.

```
1. Initialize an empty dictionary to keep track of the points of each team.
2. For each competition in the competitions list:
     1. Determine the winner of the competition (home team if result is 1, away
team if result is 0).
     2. If the winner is not in the points dictionary, add them with a score of 0.
     3. Add 3 points to the winner's score in the points dictionary.
3. Find the team with the maximum points in the points dictionary.
4. Return the team with the maximum points as the tournament winner.
```

> **Team note:** Come up with a high-level solution. This can involve brainstorming sessions with different teams, including design and product teams. Sketch out rough UI designs or system diagrams if necessary.

Step 5 - Discuss Solution Alternatives (Hows)

Discuss and evaluate potential alternative solutions to the problem.

> **Team note:** Evaluate different solution paths. Consider technical feasibility, design aesthetics, user experience, cost, time, and other factors. This stage may involve getting feedback from different stakeholders.

Step 6 - Initial Approach/Drafting

Once a solution is chosen, a prototype or draft solution is created.

> **Team note:** Start drafting the solution. For software problems, this would involve writing pseudocode or actual code. For design problems, this could involve creating wireframes or design mockups.

Step 7 - Optimization/Finalization

After testing and refining the initial draft, the solution is optimized and finalized.

> **Team note:** Refine and optimize your solution. For a design problem, this might mean making the design more intuitive and aesthetically pleasing. For a coding problem, this might involve optimizing your code for efficiency.

Step 8 - Validation

Finally, the solution is tested with various test cases to ensure its correctness and efficiency.

In the case of testing it is very good to have clear names on testing scenario for everyone to understand the intentions.

```
# Test Case 1: Basic Input ...
competitions = [["A", "B"], ["B", "C"], ["C", "A"]]
results = [1, 1, 1]
assert tournamentWinner(competitions, results) == "B"

# Test Case 2: One Team wins all matches ...
competitions = [["A", "B"], ["A", "C"], ["A", "D"]]
results = [1, 1, 1]
assert tournamentWinner(competitions, results) == "A"
```

> **Team note:** Test your solution to ensure it meets all requirements and performs as expected. This could involve unit testing for code, usability testing for designs, and getting final approval from stakeholders.

Step 9 - Implementation and Review

> **Team note:** Implement the solution and gather feedback. This could involve deploying the code, launching a new feature, or rolling out a new design. After implementation, review the solution with stakeholders to ensure it meets their needs and expectations.
>
> By involving stakeholders at each step and considering design and product aspects, you ensure the solution you develop meets the needs of the end users and aligns with the goals of your organization.

Step 10 - Psychological safety, culture and business context

Psychological safety

Psychological safety is a crucial aspect in any collaborative environment, especially in diverse teams working on complex problems. It's the shared belief that a team is safe for interpersonal risk-taking, meaning members feel comfortable expressing their thoughts, ideas, and concerns without fear of negative consequences.

In the context of a multi-team environment, promoting psychological safety can have several benefits:

- Fosters Open Communication: When team members feel psychologically safe, they are more likely to share their thoughts, ask questions, seek help, and discuss problems openly. This can lead to better problem-solving and decision-making.
- Encourages Innovation: A psychologically safe environment encourages people to take risks and come up with innovative solutions. It allows for trial and error without fear of punishment, which can drive creativity and innovation.
- Improves Learning and Growth: Psychological safety allows team members to admit mistakes and learn from them, leading to personal growth and overall team improvement.

E nhances Team Collaboration: When team members feel safe and respected, they are more likely to collaborate effectively, trust each other, and work towards shared goals.

To promote psychological safety in a multi-team environment, you can consider the following strategies:

- Promote Respect and Fairness: Encourage team members to treat each other with respect and fairness. This includes listening attentively to others' ideas, avoiding harsh criticism, and promoting equality.
- Encourage Open Communication: Create an environment where team members feel free to express their thoughts, ask questions, and share their ideas.
- Normalize Mistakes: Frame mistakes as opportunities for learning, not as failures. This can encourage team members to take risks and innovate without fear of retribution.
- Show Empathy and Understanding: Be understanding and empathetic to team members' challenges. Recognize that everyone can have off days and that personal circumstances can affect work performance.
- Lead by Example: Leaders should model the behavior they want to see in their team. If leaders show vulnerability, admit their own mistakes, and encourage open communication, team members are more likely to do the same.

Implementing these strategies can help create a psychologically safe environment where all team members feel comfortable contributing to their full potential.

Cultural safety

In a broader sense, when working in a diverse and distributed environment with many teams, there are a wide variety of factors to consider for successful management and coordation.

- Agile Practices: Implementing Agile methodologies can help in managing the complexity of working with many teams. This includes practices like Scrum, Kanban, or scaled Agile frameworks like SAFe (Scaled Agile Framework) or LeSS (Large-Scale Scrum). These methodologies can improve project transparency, adaptability, and foster a culture of continuous improvement.
- Communication and Collaboration Tools: In a multi-team environment, especially with remote teams, it's important to use appropriate tools for communication and collaboration. Tools like Slack, Microsoft Teams, or Discord can facilitate communication, while tools like Jira, Trello, or Asana can be used for project management.

- Regular Sync Ups: Have regular meetings or stand-ups to update on progress, discuss blockers, and ensure all teams are aligned towards the common goal. This helps in catching any miscommunications or misunderstandings early on.
- Cultural Sensitivity: With diverse teams, it's important to be sensitive to cultural differences. This can include understanding different work styles, communication habits, holidays, and respecting time zones for scheduling meetings.
- Clear Roles and Responsibilities: Clearly define the roles and responsibilities of each team member to avoid confusion and conflict. Everyone should understand their tasks, responsibilities, and to whom they should report.
- Documentation: Maintain up-to-date documentation of project requirements, design decisions, coding standards, and other relevant information. This is especially important when teams are working in different time zones and direct communication might be challenging.
- Conflict Resolution Mechanisms: Have a clear process in place to resolve conflicts. This could be through designated mediators, escalation paths, or through regular retrospectives where teams can air grievances and work towards solutions.
- Continuous Integration/Continuous Deployment (CI/CD): In a software development environment, implementing CI/CD practices can help to catch integration issues early and ensure smooth deployment of the software.

Remember, every team and project is unique, and what works best will depend on the specific needs and constraints of your teams and projects. Adapt these strategies as necessary to best fit your situation.

The Business Context

In a business context, while striving for psychological safety, teamwork, and effective problem-solving, it's indeed essential to consider the broader organizational constraints and factors. Here are some significant considerations:

- Budget Constraints: Every project or initiative has a budget that must be adhered to. It's important to understand these financial constraints and manage resources effectively. This might involve prioritizing certain tasks, finding cost-effective solutions, and continually monitoring and adjusting financial plans.
- Legal and Regulatory Requirements: Businesses operate within a legal and regulatory framework that can impose constraints on how they operate. Compliance with these laws and regulations is non-negotiable. It's important for teams to understand these requirements and ensure their solutions are compliant.
- Corporate Culture: The culture of the organization can significantly influence how teams work together. For instance, a company with a hierarchical culture might have a different decision-making process than one with a flat structure. Understanding and navigating these cultural aspects is important for effective collaboration and problem-solving.
- Time Constraints: Deadlines are a common aspect of any project. Teams need to manage their time effectively to ensure that they meet these deadlines. This might involve breaking down tasks, setting clear milestones, and regularly monitoring progress.

- Competitive Environment: The competitive landscape can influence business decisions and project priorities. Understanding the competition can help inform strategy and decision-making.
- Stakeholder Expectations: Different stakeholders might have different expectations for a project. Balancing these expectations and managing stakeholder relationships is a key part of any project.
- Resource Availability: The availability of resources, including human resources, tools, technology, and materials, can impose constraints on what's achievable.
- Risk Management: Businesses face various risks, from financial and operational risks to strategic and reputational risks. Effective risk management is important to identify, assess, and manage these risks.

Understanding these constraints and factors, and learning how to navigate them, is a key part of working in a business environment. It involves not just technical or project management skills, but also business acumen, interpersonal skills, and the ability to balance different priorities and make informed decisions. It's a complex process, but with experience and learning, teams can effectively operate within these constraints to achieve their goals.