International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

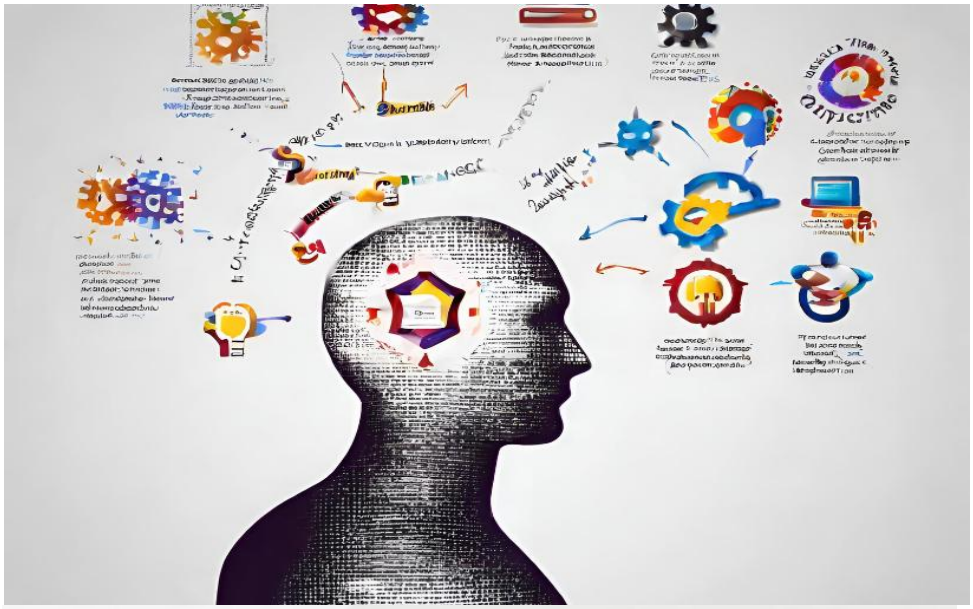
Application of Personification Thinking to Achieve Higher Software Quality

Ravikiran Mallappa

QA Manager, Fourth frontier Technologies Pvt Ltd, 523 Nelligudde Tank Road, Bidadi, 562109, Karnataka, India
Email: *ravikiranhm07[at]gmail.com*

Abstract: ***This article explores the application of personification thinking in software development life cycle (SDLC) to enhance software quality. By personifying each phase of SDLC we can increase the quality by reducing the number of iterations in each phase due to low quality and eliminate the different types of Muda [1] (waste) early in the SDLC***

Keywords: Personification thinking, User stories, User centric design process, QA Metrics, SDLC, Muda elimination, Ease of communication, test planning.



1. Introduction to Personification Thinking

Personification [2] is a type of figurative language in which human characteristics are given to non-human things. This can include physical characteristics, emotions, thoughts, or actions.

Personification is a powerful tool for communication because it can help us to create vivid and engaging images in the minds of our readers or listeners. It can also be used to express complex ideas in a more accessible way [3]. For example, a writer might say that "the economy is growing" or that "the stock market is on fire." This helps us to visualize and understand abstract concepts in a more concrete way.

Why do we need personification thinking?

Personification thinking can be used in a variety of ways in writing and speech. For example, it can be used to:

- **Express complex ideas in a more accessible way**. Personification can be used to explain complex concepts in a way that is easier to understand. For example, we might say that "the economy is growing" or that "the stock market is on fire." This helps us to visualize and understand abstract concepts in a more concrete way.
- **Add humor and interest to writing or speech**. Personification can be used to add humor and interest to our writing or speech. For example, we might say that "the sun is smiling down on us" or that "the wind is whispering through the trees." These personified phrases can make our writing more engaging and memorable.
- **Develop a character's personality or perspective**. Personification can be used to develop a character's personality or perspective. For example, a writer might personify the character's conscience as a voice in their head that is constantly giving them advice. This can help the reader to understand the character's thoughts and feelings in a more personal way.

How to Apply Personification Thinking in SDLC

Here are some ways to apply personification thinking in each phase of software development:

a) **Requirements gathering**: Personification can be used in the requirement-gathering phase of the SDLC [4] to help different stakeholders understand and visualize the needs of the users. For instance,

b) When writing user stories, try to focus on the user's goals and motivations. What are they trying to achieve by using the system? What are their needs and pain points? Example,
   - As a [user role], I want to [do something] so that [I can achieve something].
   - As a customer, I want to be able to add items to my shopping cart so that I can purchase them later.
   - Use active voice. Active voice is more engaging and easier to read than passive voice. When writing requirements, try to use active voice whenever possible.
   - For example, instead of writing "The system shall allow users to create and edit profiles," you could write "Users can create and edit their profiles."

c) Personification can help you to write more user-centered requirements. By using personas, user stories, and active voice, you can make your requirements more relatable and understandable to stakeholders which help in reducing irritations and removing ambiguities in requirement review phase and increase the quality of requirement

d) **Design:** Personification thinking can be applied to the design phase of SDLC in a number of ways. Here are a few examples:
   - **Personify the target user**: One way to ensure that your design is user-friendly is to personify the target user and keep them in mind throughout the design process. This means thinking about their needs, wants, and frustrations. You can create a user persona to help you do this. For instance,
   - Designing a new e-commerce platform: When designing a new e-commerce platform, you could personify the target user as a busy shopper who wants to be able to find what they need quickly and easily. You could then design the platform with this user in mind, making sure that it is easy to navigate and that the products are well-organized.
   - **Use personification in the design itself**. You can use personification in the design itself to make it more engaging and memorable. For example, you could use personified characters or icons to represent different features of the system. You could also use personified language in the user interface.
   - Personification can help you to create user centric design [5]. By Early feedback from users on personified design a team can reduce weakdesign

work and increase the stability of design early in the phase.

e) **Implementation:** By applying personification thinking to your code, you can make it more readable, understandable, and maintainable so that any other person who's going to use the code to review, enhance, or maintain will not feel clumsy. By following a few of the tips given below we can apply personification thinking to our code which will add quality to what we are doing.

- *Give meaningful names to variables and functions*. This will make your *code more readable and easier to understand*. For example, instead of using a variable name like x, use a variable name that describes what the variable is actually used for, such as customer_name or order_total.
- *Organize your code into logical blocks*. This will make your code easier *to read and maintain*. For example, you can use functions to group related code together, and you can use classes to encapsulate data and behavior.
- *Use whitespace effectively*. Whitespace can help *to improve the readability*of your code. For example, you can use indentation to show the logical structure of your code, and you can use blank lines to separate different sections of code.
- By applying personification thinking to your code, you can make it more readableunderstandable and clean [6]this helps during code review phase which in identifying issues in code which in turn helps redundant implementation work which eventually increase the code quality.

f) **Testing:** Testing phase includes test planning, execution and reporting the status to different stakeholder. We can apply personification in to planning by

- *Personify the test objectives*. One way to make test objectives more specific and measurable is to personify them. This means giving them names and personalities. For example, you could have a test objective called "The Happy Customer" or "The Productive Sales Representative." This can help you to focus on the needs of the users and to ensure that the test plan is designed to meet those needs.
- *Personify the test environment*. Another way to use personification thinking in test planning is to personify the test environment. This means thinking about the test environment's capabilities, limitations, and goals from end users' point of view. You can create a test environment persona to help you do this. This can help you to design a test plan that is realistic and achievable.
- *Personification reporting:* While sending out test report, stand in the shoe of the stakeholder who's expecting the report and build that specific report to stakeholder with key QA metrics [7]which adds value to report
- By applying personification thinking in test design and planning phase team can reduce defect escape rate and better test coverage which will ensures quality product to end users.

g) **Maintenance:**

- Here are some specific examples of how personification thinking can be applied in the maintenance phase of SDLC:
- Maintaining a new e-commerce platform: When maintaining a new e-commerce platform, you could personify the system as a customer-facing system that needs to be reliable and available at all times. You could then develop maintenance activities that minimize downtime and ensure that the system is able to handle peak traffic loads. You could also personify the users as busy shoppers who want to be able to find the products they need quickly and easily. You could then develop maintenance activities that improve the performance of the search engine and the checkout process.
- Maintaining a new customer relationship management (CRM) system: When maintaining a new CRM system, you could personify the system as a tool that helps sales representatives to be more productive. You could then develop maintenance activities that fix bugs and add new features that will help sales representatives to close more deals. You could also personify the users as sales representatives who need to be able to access their data from anywhere and at any time. You could then develop maintenance activities that improve the mobile app and the cloud-based interface.

Key Takeaway and Conclusion

Personification thinking has a number of benefits in the SDLC, including but not limited to

- Ensure that the software is user-centered.
- Identifying and address potential usability problems early on in the development process.
- Improved communication
- Increased understanding
- Reduced risk
- Improve the overall quality of the software
- Reduce development cost.

References

[1] https://www.process.st/muda/
[2] https://leverageedu.com/blog/personification/
[3] https://tara.sdu.edu.tr/vufind/Record/EBC918226/TOC
[4] https://www.synopsys.com/glossary/what-is-sdlc.html
[5] https://www.usability.gov/what-and-why/user-centered-design.html
[6] https://ptgmedia.pearsoncmg.com/images/9780132350884/samplepages/9780132350884.pdf
[7] https://www.testrail.com/blog/qa-metrics-matter/