Making Implicit Concepts Explicit

Task

Follow the slides and the chapter "Supple Design" in [1]. Find examples in your project (domain) where you can apply the patterns of this chapter. Be sure to reflect all such changes in the model, do not let the model and the code diverge.

1. Intention-revealing interfaces: use names to reveal effect and purpose (e.g. better introduce a named intermediate variable if that allows you to remove a comment), make the source code speak about intentions, not about means.

2. Side-effect free functions: review your functions for side-effects. Is the side-effect combined with a calculation, computation, or query? Separate them. Can you turn the side-effect on an object state into a value object?

3. Does the model appeal to a users intuition?

4. Consider assertions (pre-/post-conditions, class invariants, aggregate invariants) to capture any remaining side-effects.

5. Reflect on which conceptual contours you have gradually made visible that were initially unclear.

6. Where can you apply stand-alone classes to reduce the cognitive load in understanding the model?

7. Can you find opportunities where closure of operations simplifies the model?

References

[1] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, Boston, 2004.