



The Design is a Projection

Gerhard Esterhuizen

Stone Three Signal Processing

Cape Town SPIN 5

17 August 2005



Introduction

- Analysis at the code level
- My definition of design
- Where design adds value
- My views on “The Code is the Design”



Characteristics of Code

- Easy to execute
 - Ordered
 - Unambiguous
 - Localised
- Grouped by
 - Abstraction
- Provides a language level view



What Code is Good For

- Execution using an FSM
- Language level
 - Implementation
 - Analysis
 - Verification
- Debugging local errors



What Code is Bad For

- Inferring
 - Broad system operation
 - Subsystem responsibilities
 - The underlying model
- Intuitively grasping complex things
 - Multithreading
 - Interdependencies and hierarchies
 - Deadlock
 - Race conditions

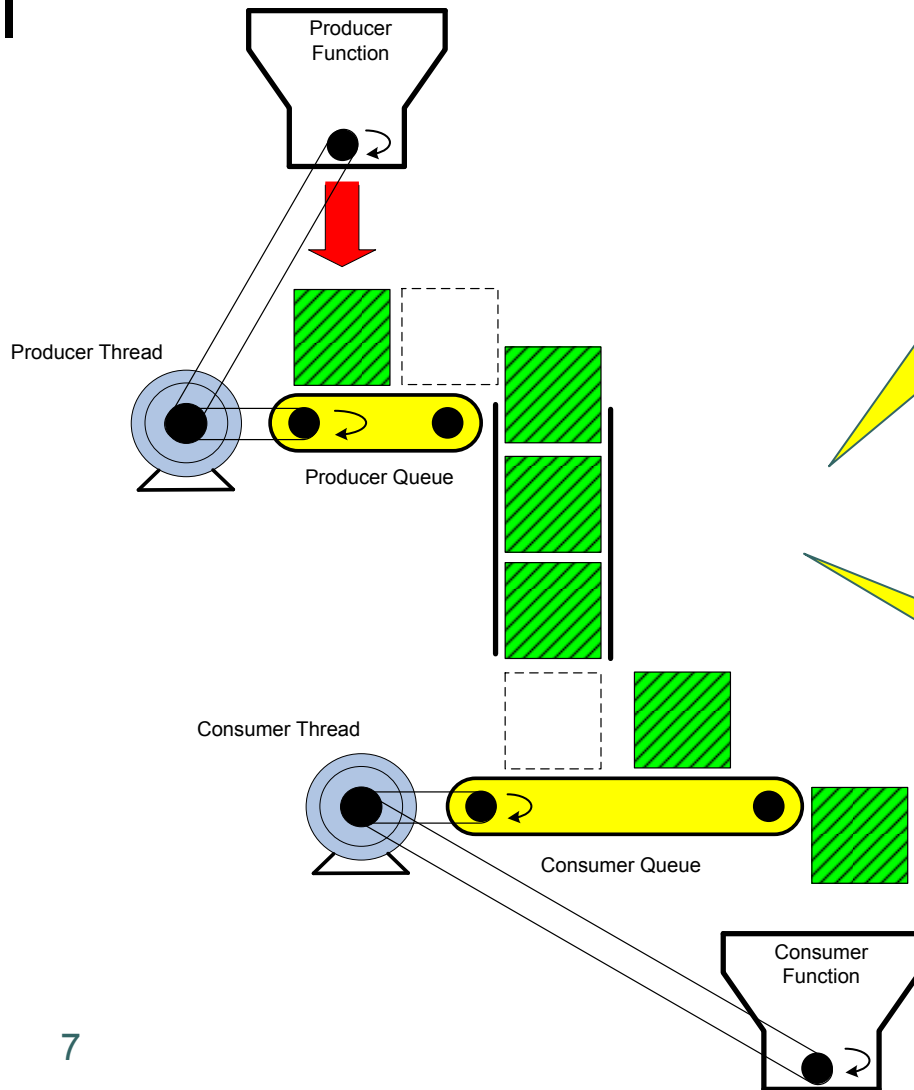


So, What is Design ?

- Design is a projection of the code
 - Optimised towards viewing a certain aspect
 - Allows analysis at an intuitive level
- The level at which you ideally would want to think
- Design space is defined in terms of abstractions
 - Disregard information deemed to be immaterial
- A design can represent
 - Already existing code
 - Code in progress
 - Just a concept

Careful: It is paramount that the abstraction acknowledges the realities of the implementation language.

Producer / Consumer: Design



Even your Grandmother can understand this...

Please don't take this too literal – it is an extreme example.



Producer / Consumer: Code

```
# Producer Thread
while True:
    o = produce()
    queue.push(o)
```

```
# Consumer Thread
while True:
    o = queue.pop()
    consume(o)
```

```
# A Blocking FIFO Queue
class Queue:
    def Queue:
        sem = Semaphore(Queue.LEN)
    def push(o):
        sem.wait()
        buffer.push(o)
    def pop(o):
        o = buffer.pop()
        sem.post()
```

Figuring this out takes a bit of a head scratch...the operation is not readily apparent, but requires careful analysis.



Ok, What's My Point ?

- The diagram and the code contains the same truths
- The diagram is easier to follow than the code
- The diagram represents a projection of the code
 - Focuses on the thread interaction
 - Illustrates the blocking mechanism intuitively



The Relation to Design

- The diagram is the design
 - It maps directly from the code
 - If the code changes the design changes

- There are more examples where the code does not provide the easiest way of looking at the problem...



Example: Multiple threads

- Model as a number of communicating FSMs
 - Consider events, state transitions and side-effects
 - Intuitive and easy to simulate
 - Coding is “soldering”
- Implement in terms of higher level constructs
 - Receive events
 - Dispatch to state-specific handler
 - Perform the side-effect
 - Perform the transition
 - Send events
- Use an implementation pattern



Example: Deadlocks

- Graph lock holders and lock takers
 - Be mechanical about it
 - Don't take shortcuts
- Solve the problem in the design
 - Establish implementation principles
- Solve the problem in the architecture
 - Employ a deadlock-free architecture



Where Does UML Fit In ?

- UML is just one type of visual model
 - Class and object hierarchies
 - Finite state machines
 - Threaded control flow
 - How groups of methods and objects interact in performing a specific task
- We often throw away large parts of our UML models
 - Code changes
 - Don't panic - modelling still shapes the result



Conclusion

- Design provides a perspective on the code which is often easier to deal with than the code itself
- Where possible, problems should be solved in the design space
 - It's often easier
 - It saves writing horribly wrong code



Conclusion

- Observe the following in design space:
 - Respect the implementation language
 - Observe strong semantics
 - Think in terms of
 - Responsibilities of modules or functions
 - The underlying conceptual model
- For an existing implementation
 - The design is a projection of the code
 - Hence, the code *is* the design.