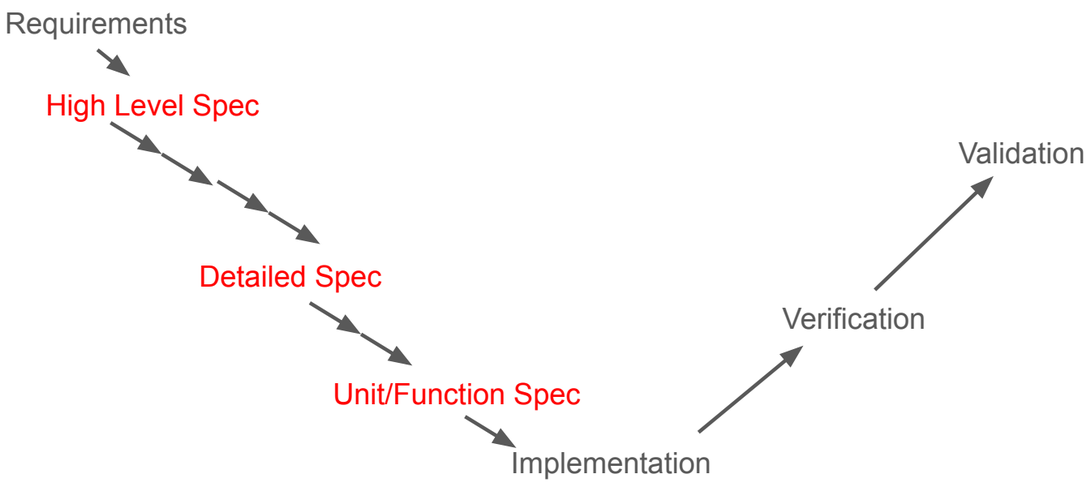
COM4506/6506: Testing and Verification in Safety Critical Systems

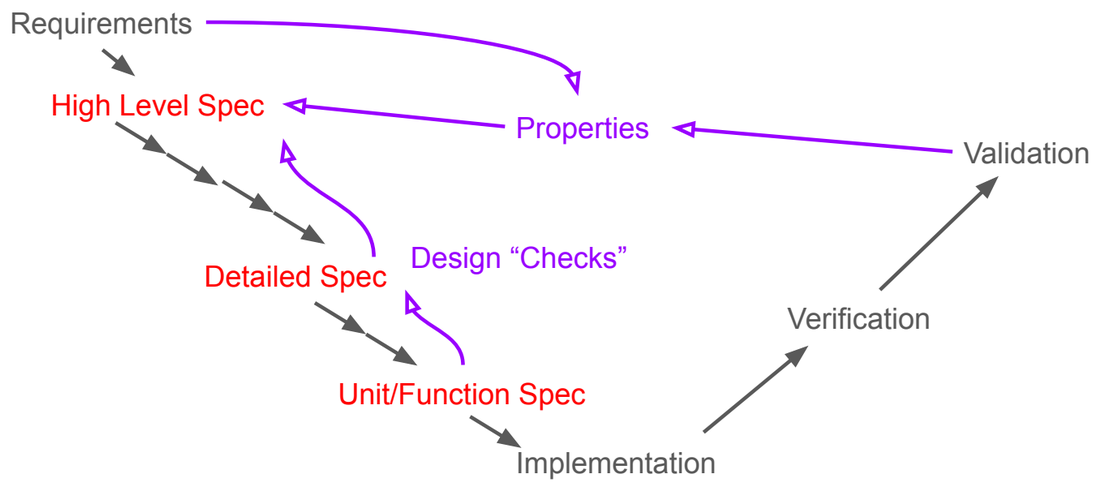
Dr Ramsay Taylor

Contents

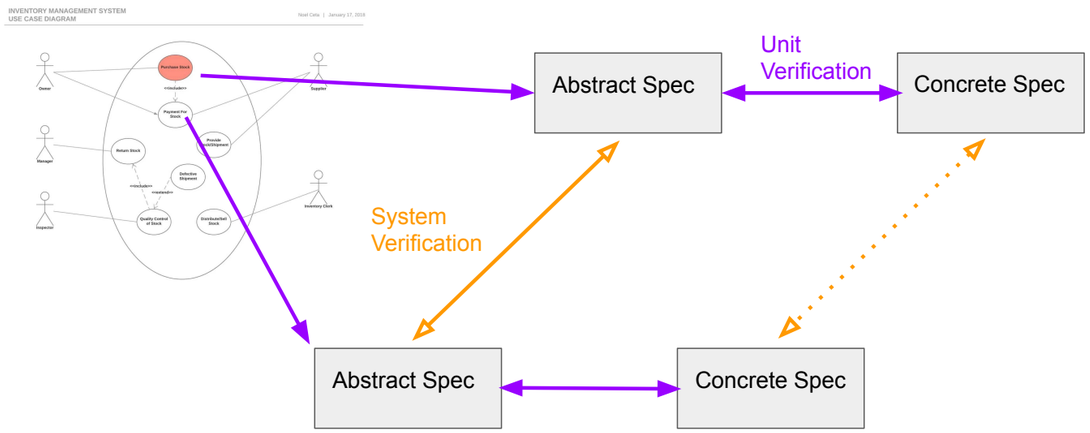
Iterative Specifications



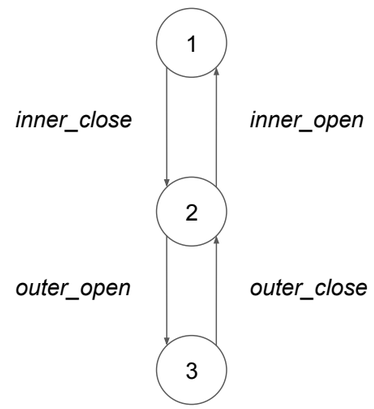
Iterative Specifications



System Components and Units



Formal Properties

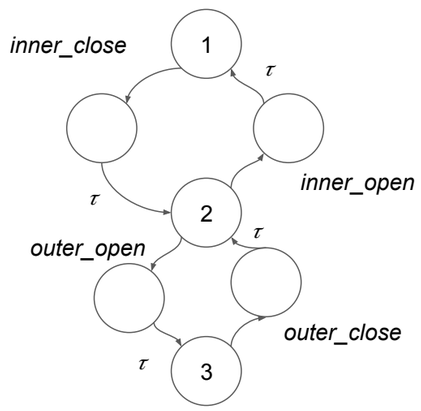


Linear Temporal Logic is one language for *expressions* of properties.

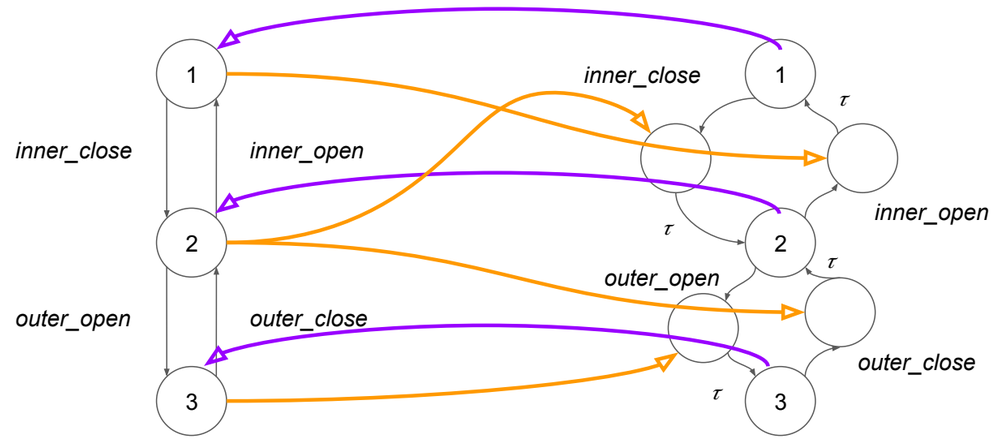
□(inner_open → ¬outer_open U inner_close)

Regular Expressions are another way to describe *traces of a system*. We will come back to that in a minute!

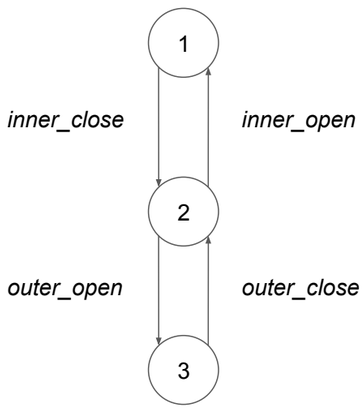
Formal Properties



Formal Properties



Refinement



Where one formal specification a more detailed version of another, we can call it a *refinement*.

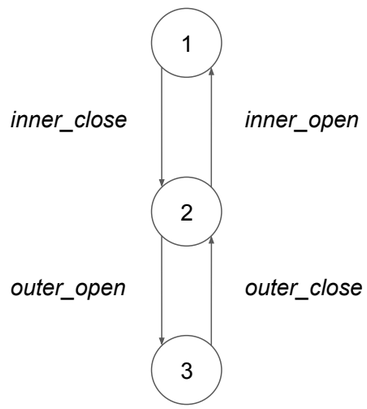
There are various mathematically defined notions of *refinement*.

One is *trace refinement*.

traces(P) = {<>, <inner_close>, <inner_close, outer_open>...}

traces(P) ⊆ traces(Q)

Refinement



Systems are defined as much by what they *can't* do as what they can - especially where we are interested in safety or security!

There are notions of *Failures Refinement*

failures(P) =

{(<>,{outer_open, outer_close, inner_open},

(<inner_close>,{outer_close,inner_close}),

...

)}

Refinement

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

There are also *State and Operation Refinement* concepts.

These generally involve *Reducing Non-determinism*

This is usually *predictable* and doesn't break the *specified properties*

… but that doesn't make it perfect!

Summary

- *Design* is an iterative process of more and more detailed *Specifications*
- We can make those specifications formal and then do various *comparisons* to check that the more detailed specifications *conform* the the higher level ones.
- *Refinement* (in various forms) is one method for handling this.
- There are others (LTL, …)