@odinthenerd

– not the god

A possible future of embedded development

Sean Parent

Everything is Crap!

A possible future of embedded development

Are you a hard real-time system?

Are you a hard real-time system?

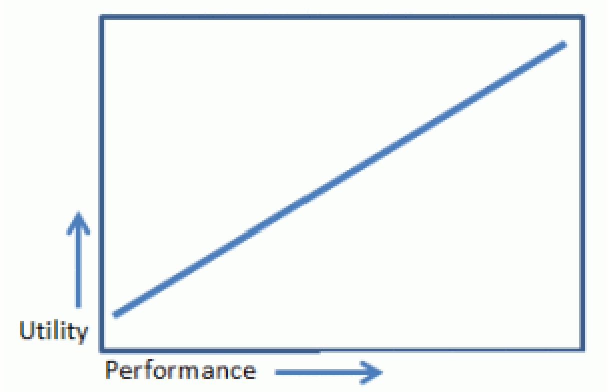
Are you master of everything?

Are you a hard real-time system?

Are you master of everything?

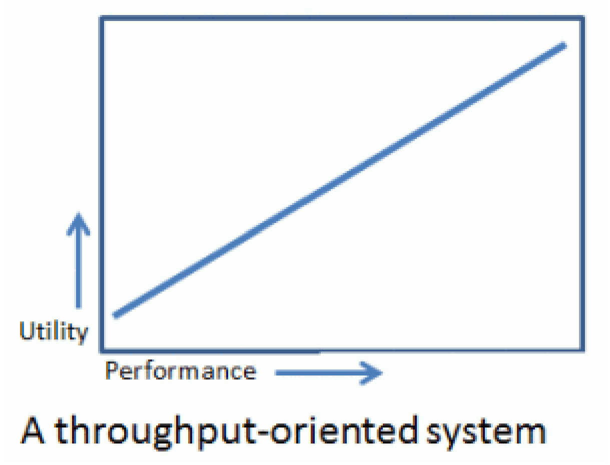
Do you assume endless RAM?

Realtime systems

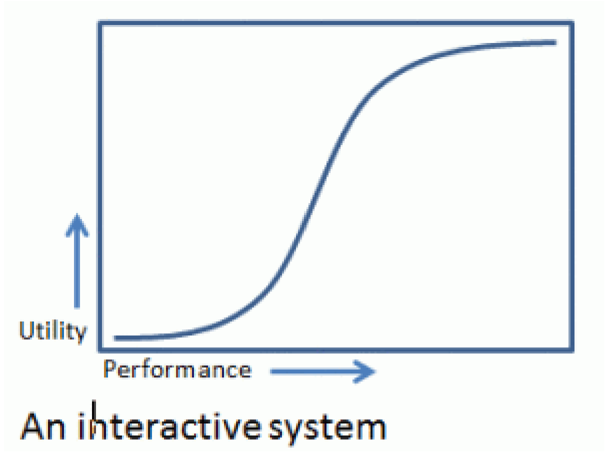


A throughput-oriented system

Realtime systems

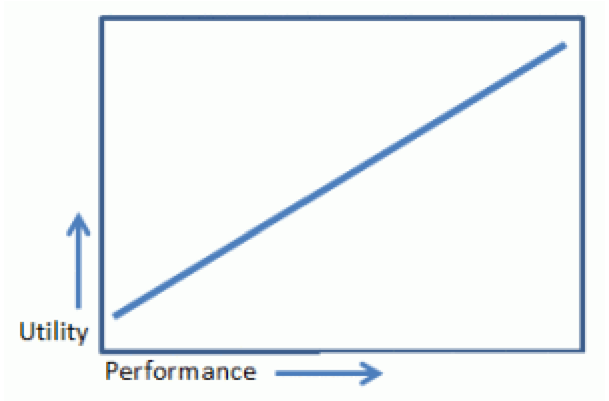


A throughput-oriented system

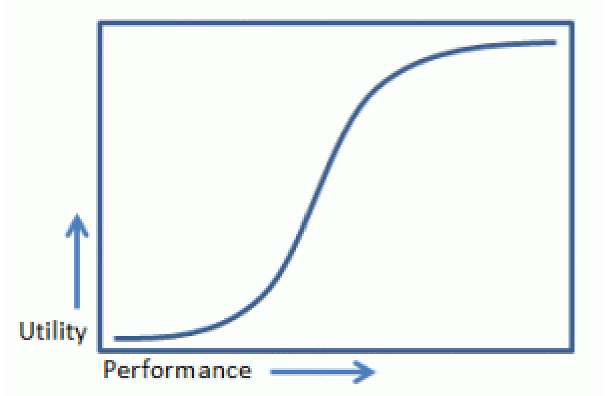


An interactive system

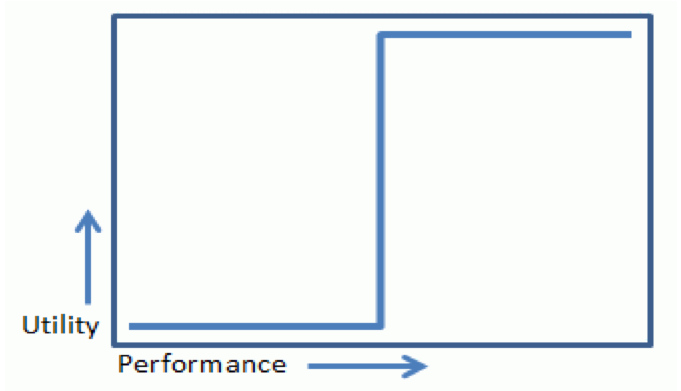
Realtime systems



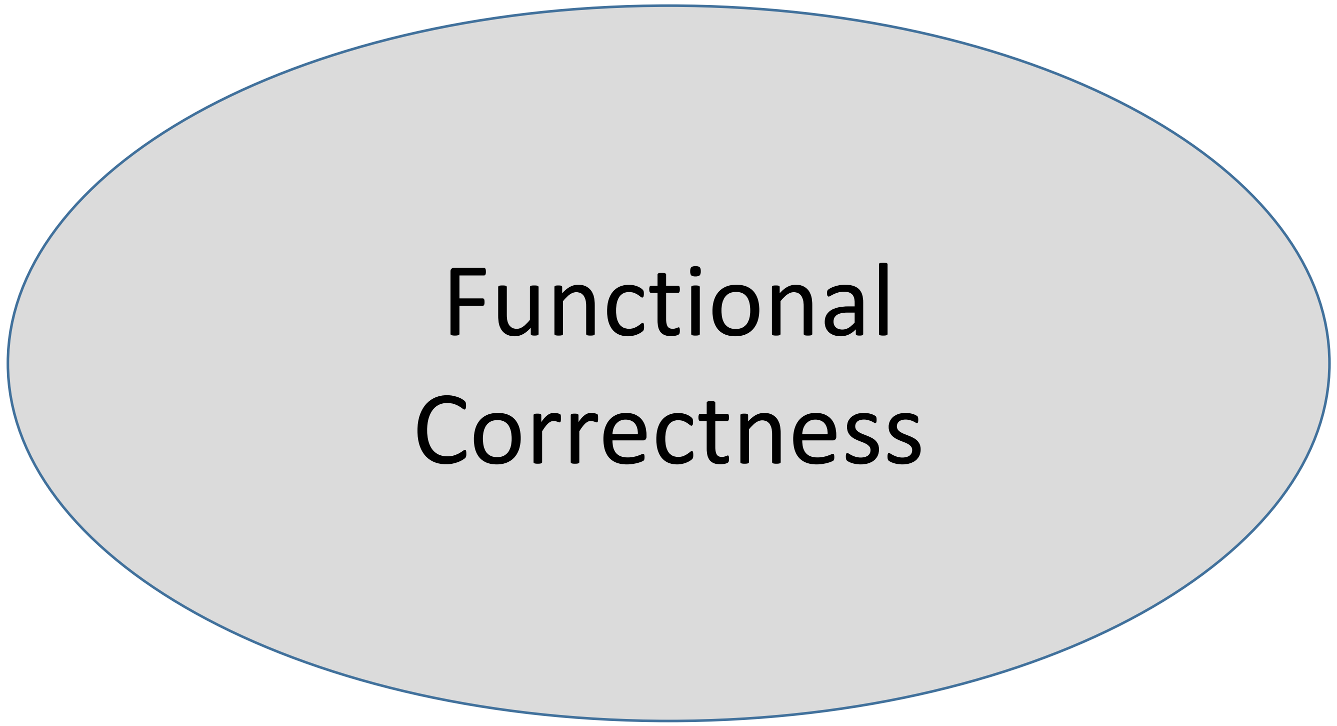
A throughput-oriented system

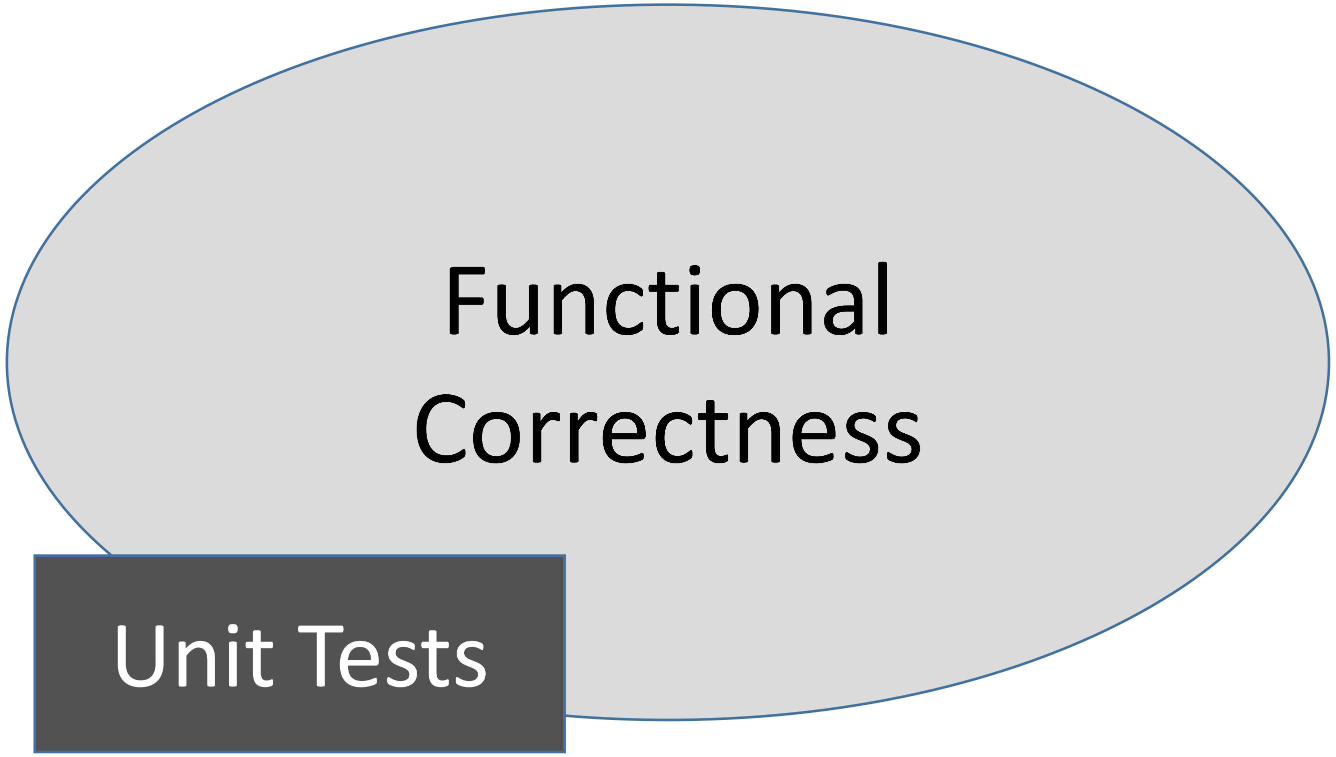


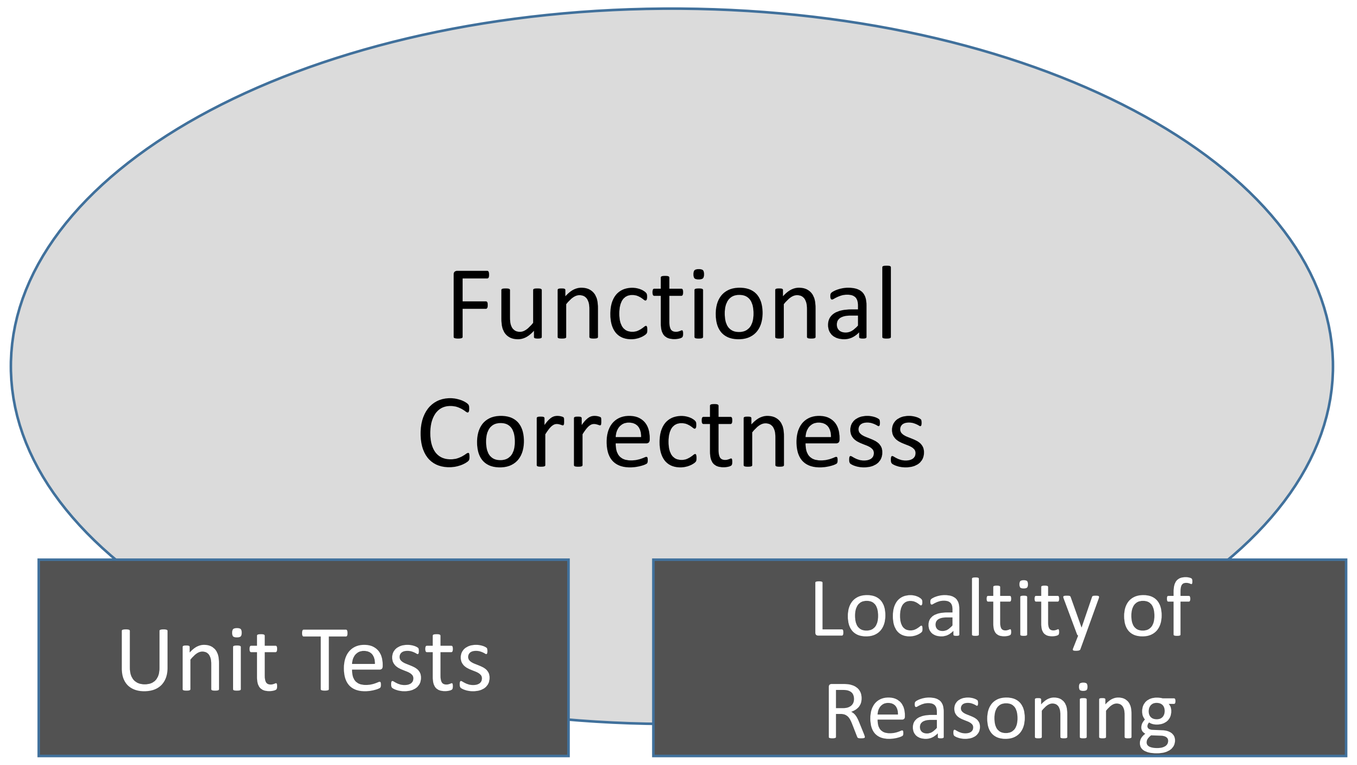
An interactive system

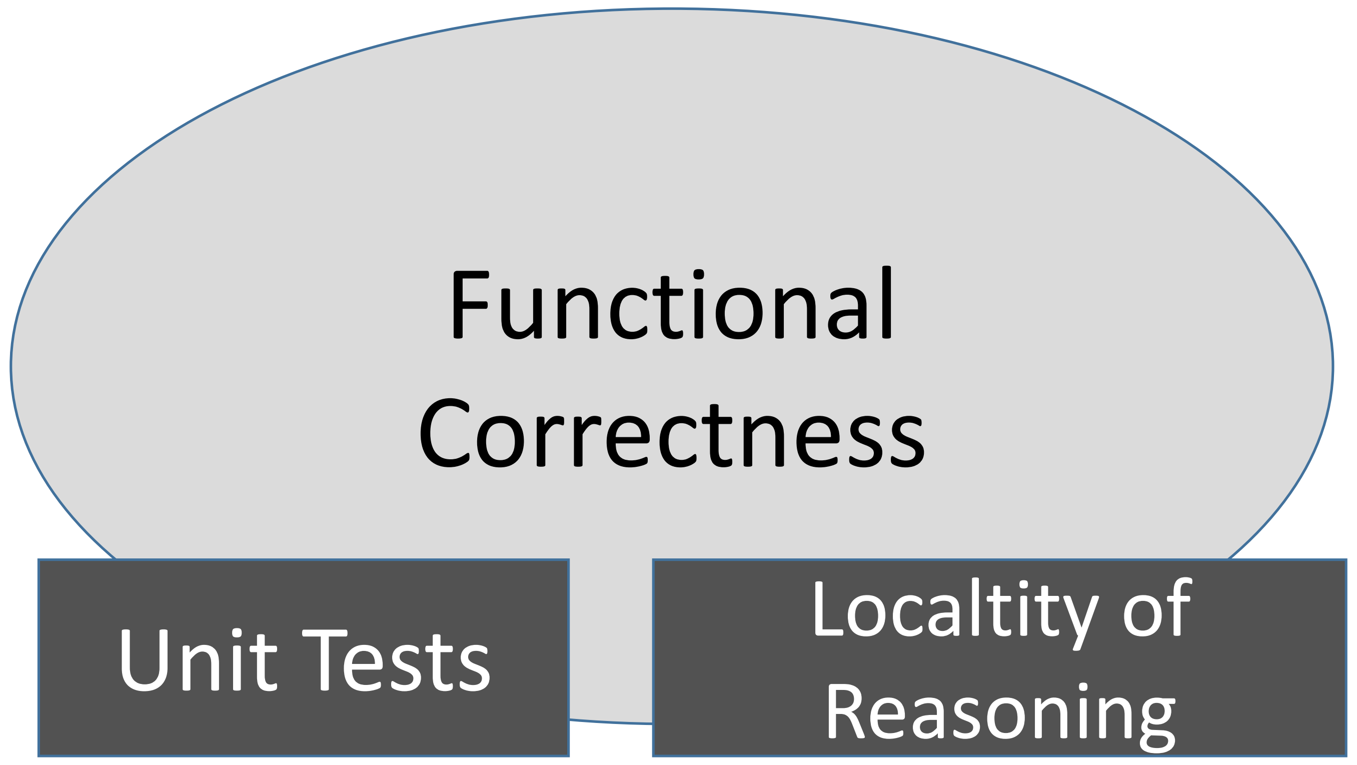


A real-time system

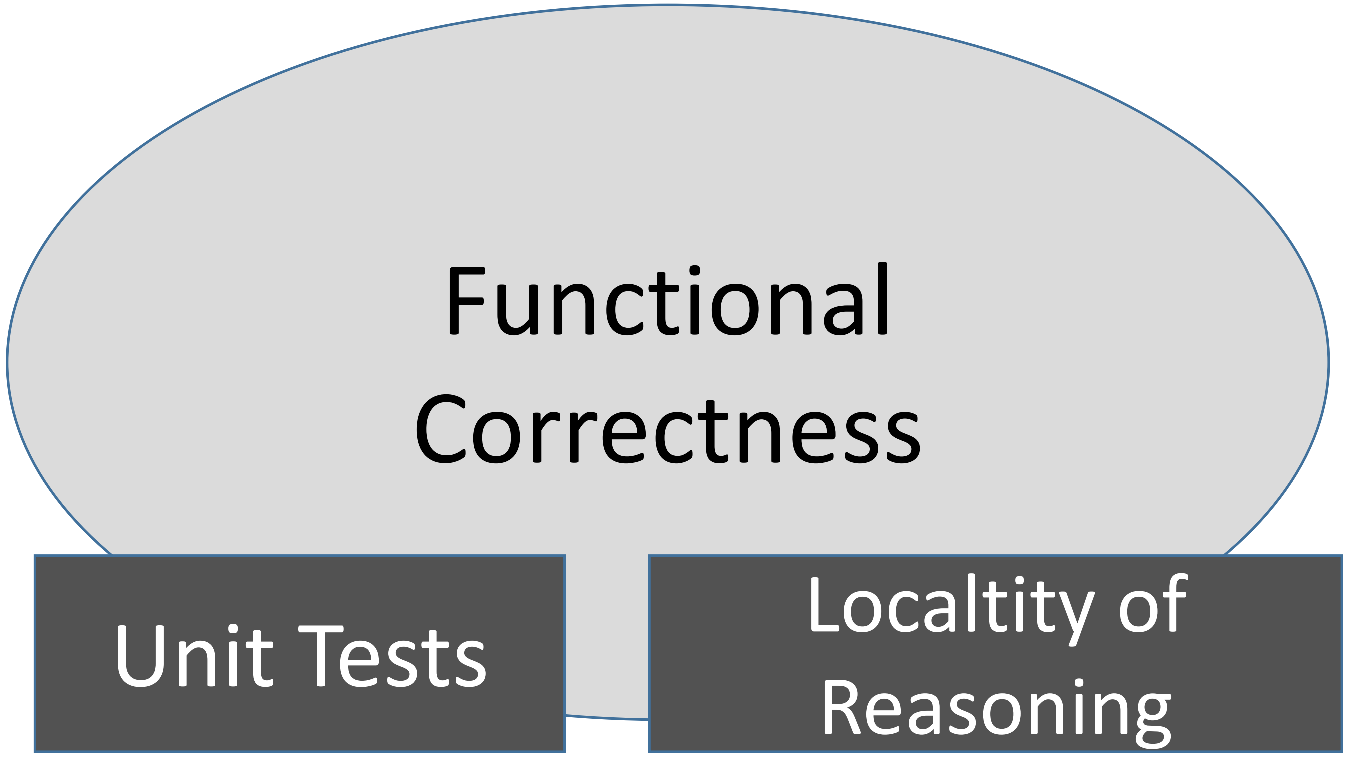

Functional
Correctness


Functional Correctness
Unit Tests

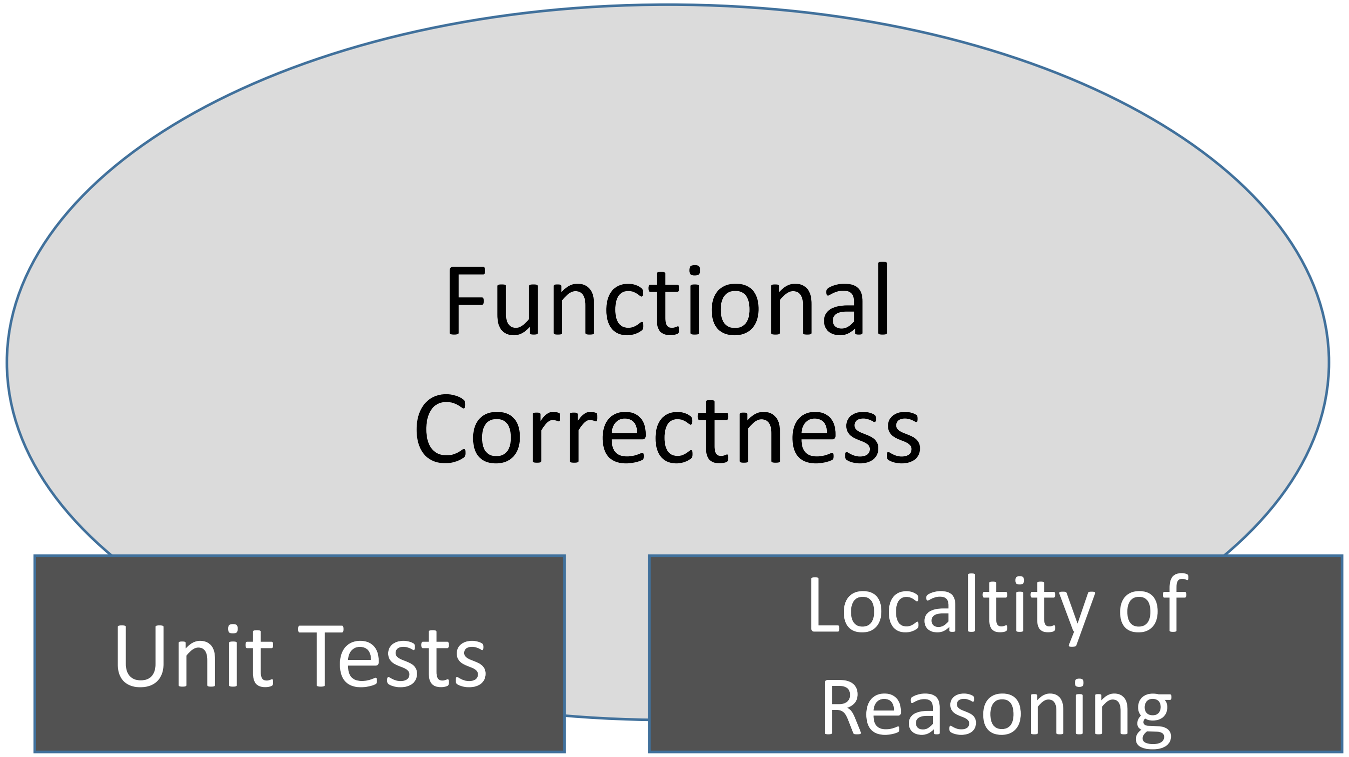

Functional Correctness
Unit Tests
Localtity of Reasoning



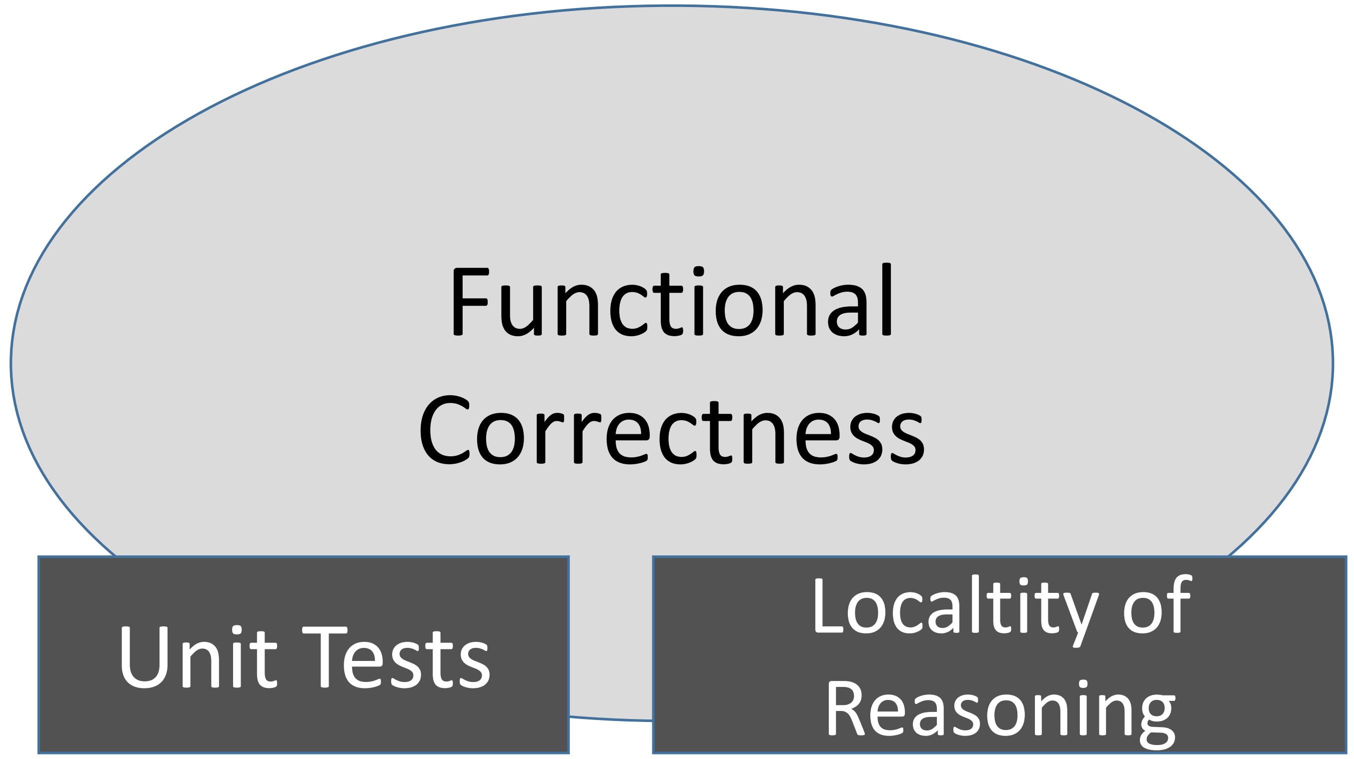
```
auto diego = std::make_unique<radioactive_frog>(charge::max_plad);
```



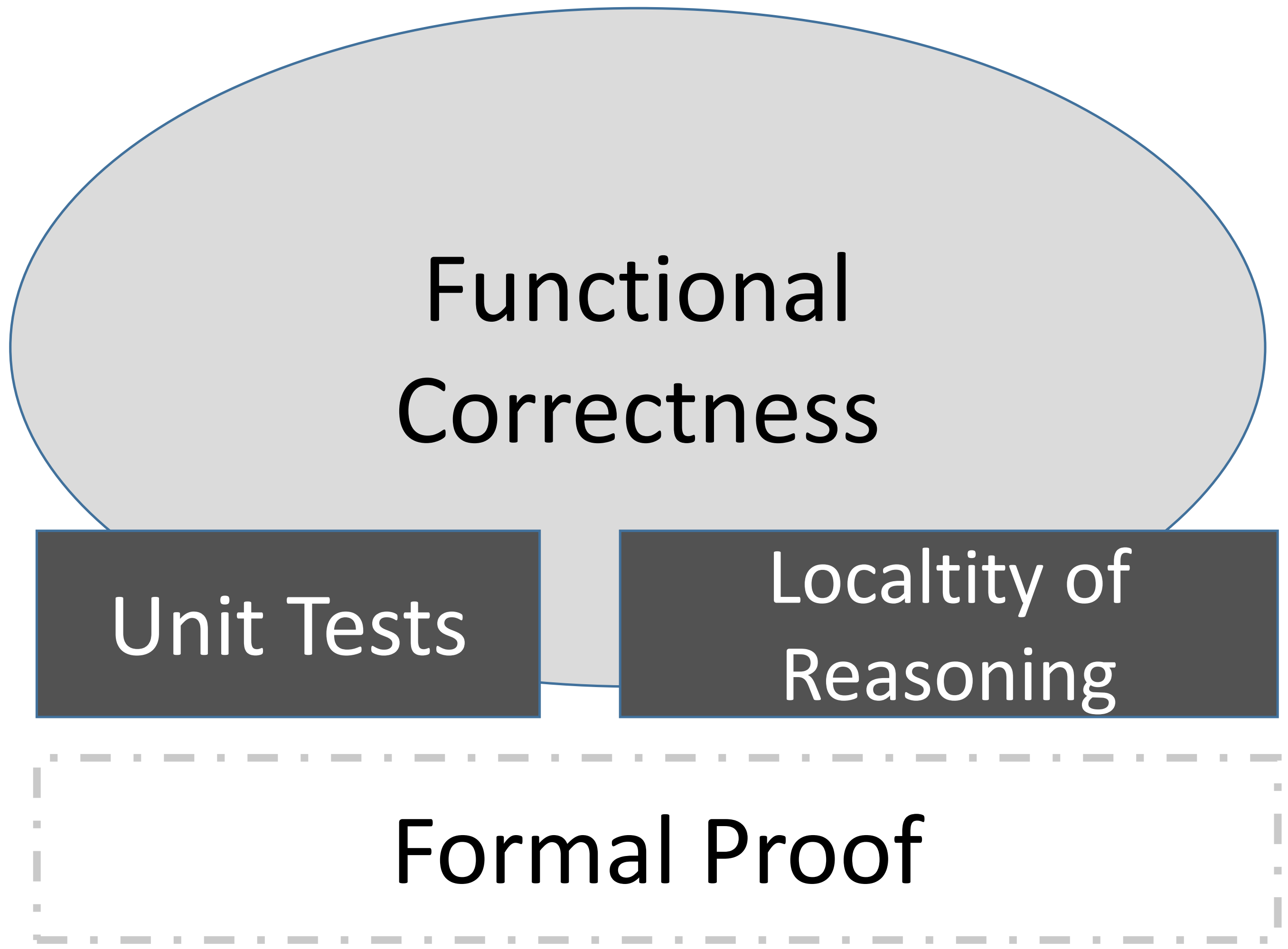
```
{
    auto diego = std::make_unique<radioactive_frog>(charge::max_plad);
}
```

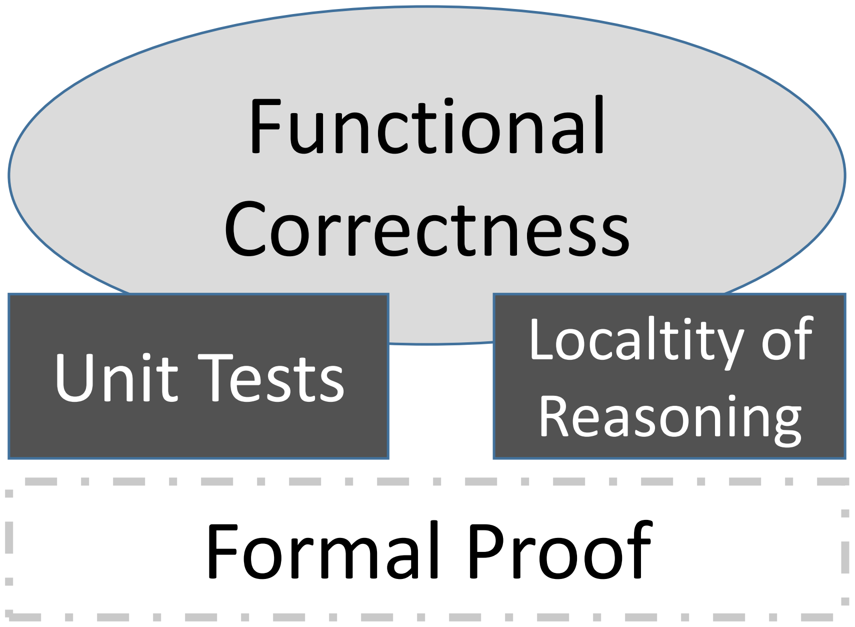


```
{
    auto diego = std::make_unique<radioactive_frog>(charge::max_plad);
    // lots of terrible things
}
```

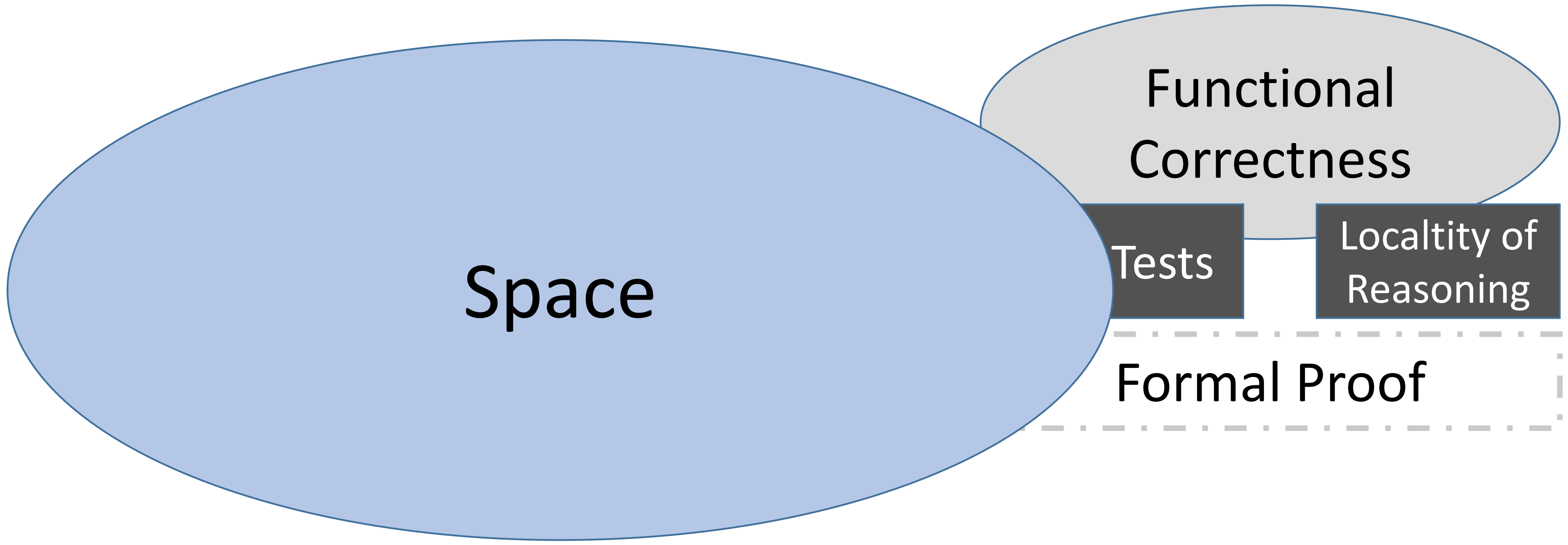


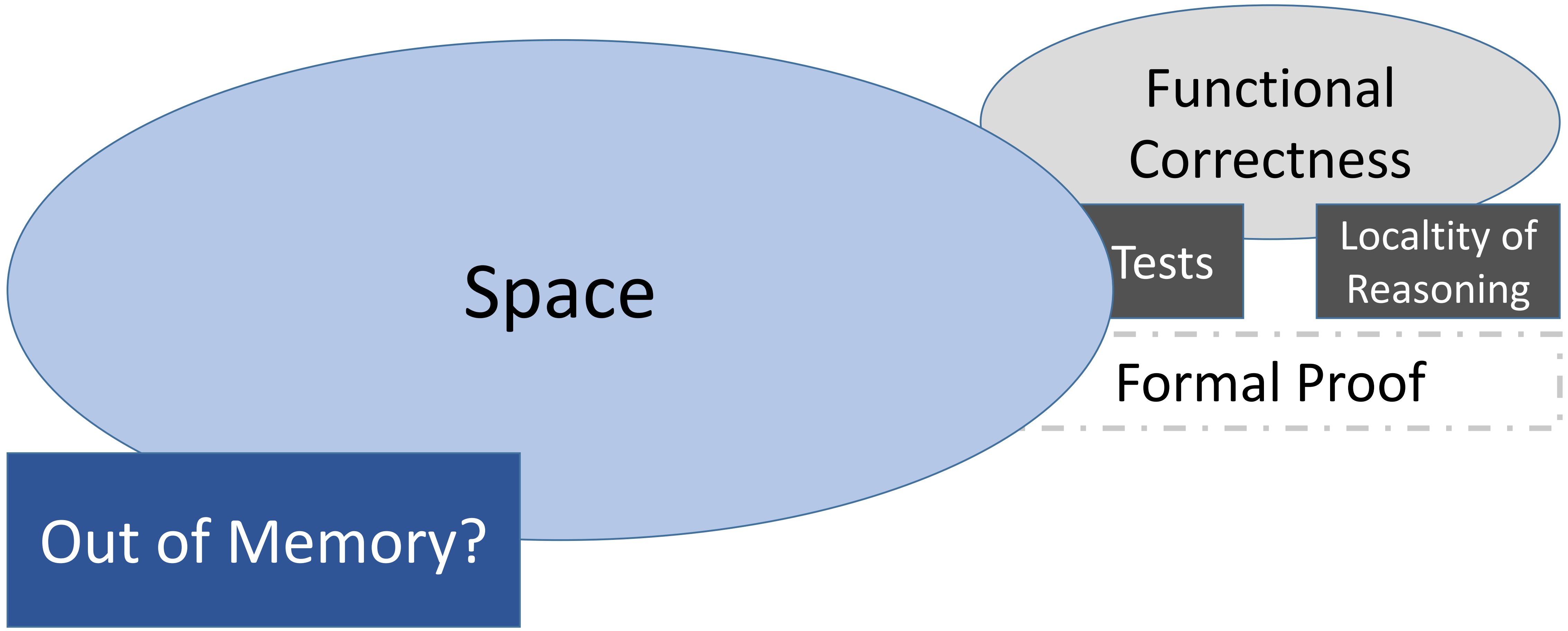
```
{
    auto diego = std::make_unique<radioactive_frog>(charge::max_plad);
    // lots of (moderately) terrible things
}
```

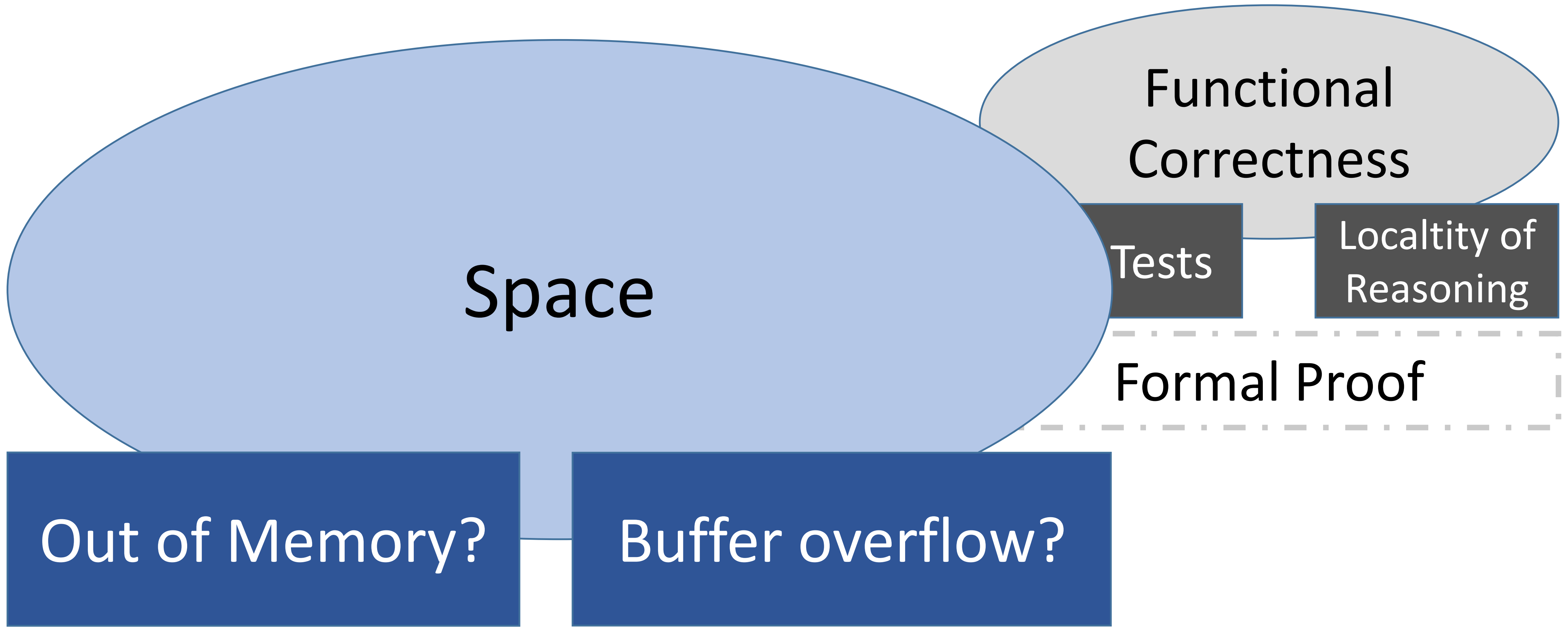

Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof

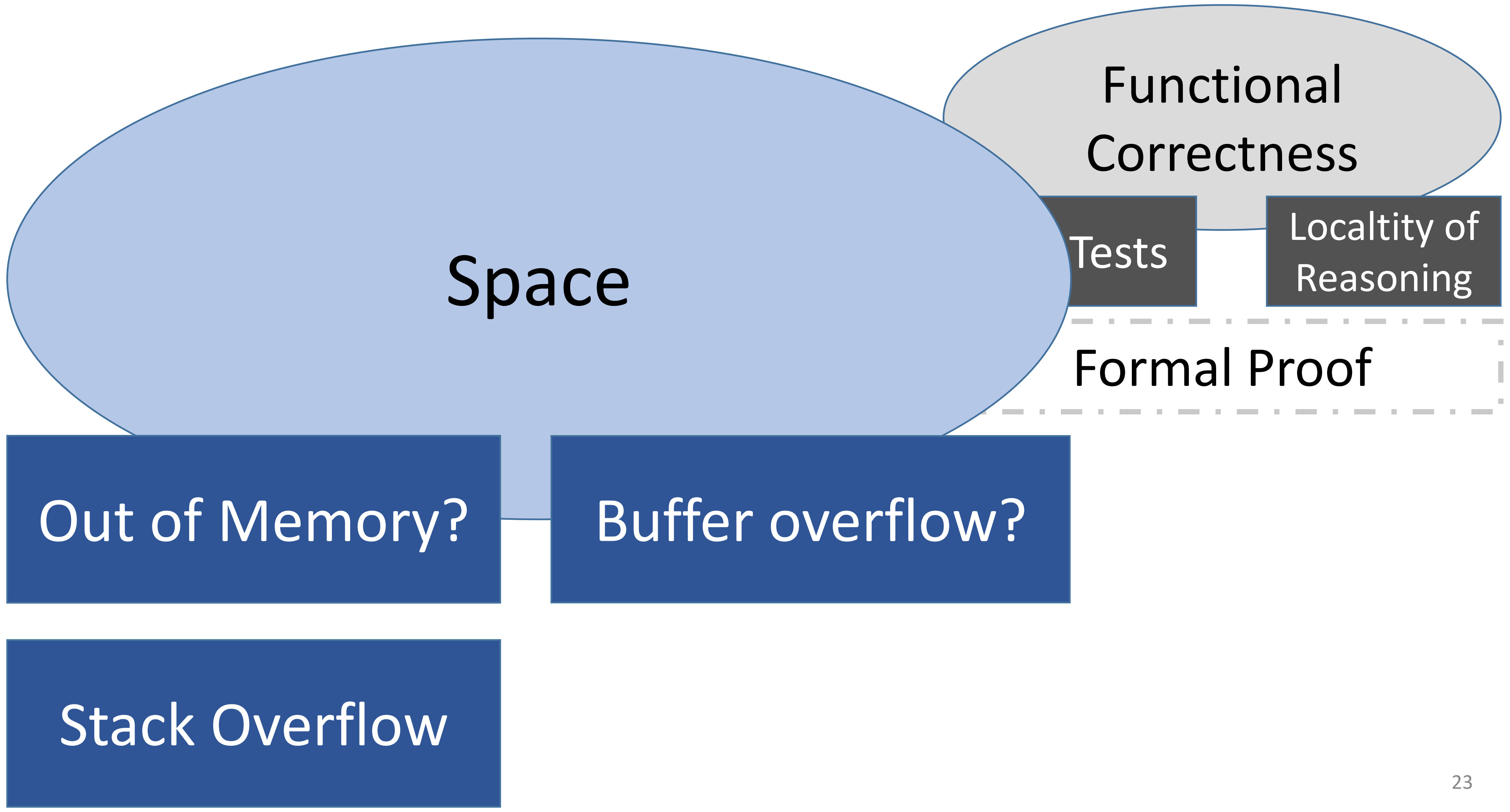

Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof

@odinthenerd

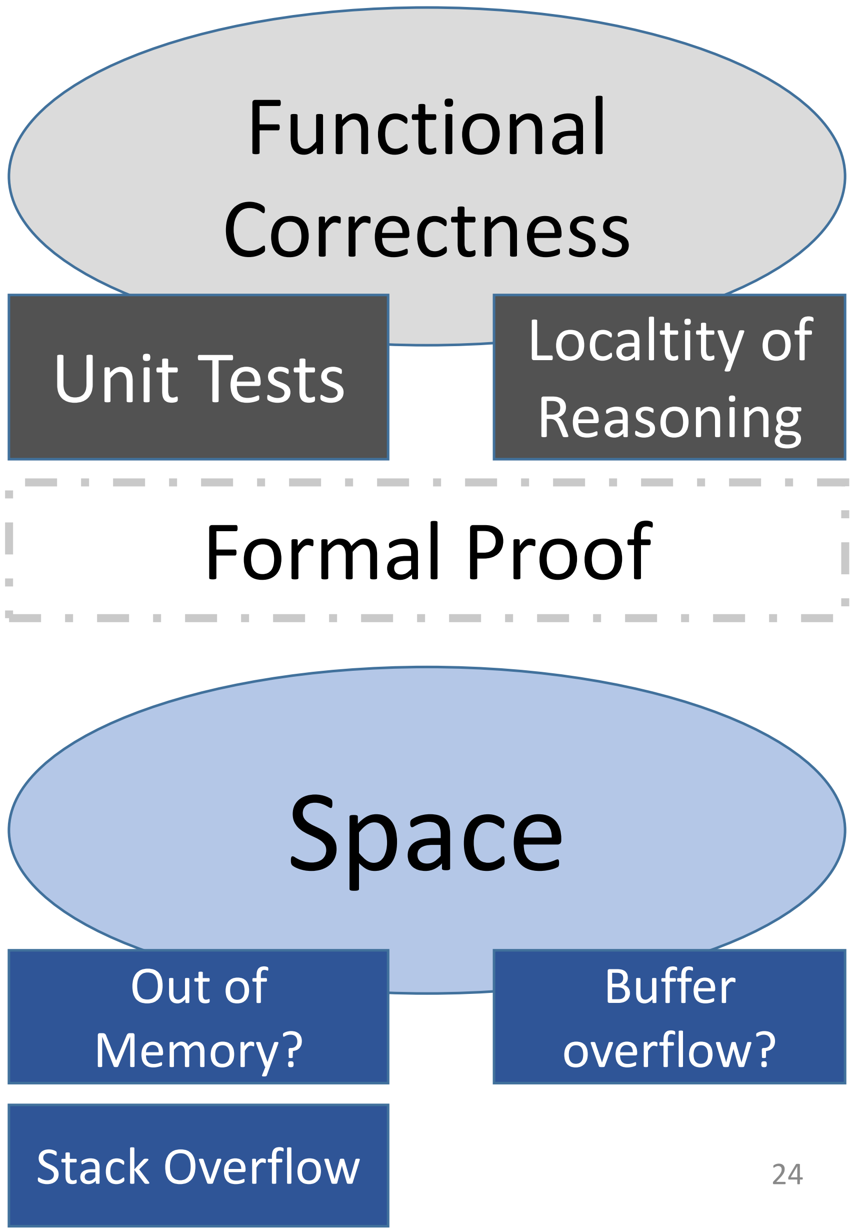


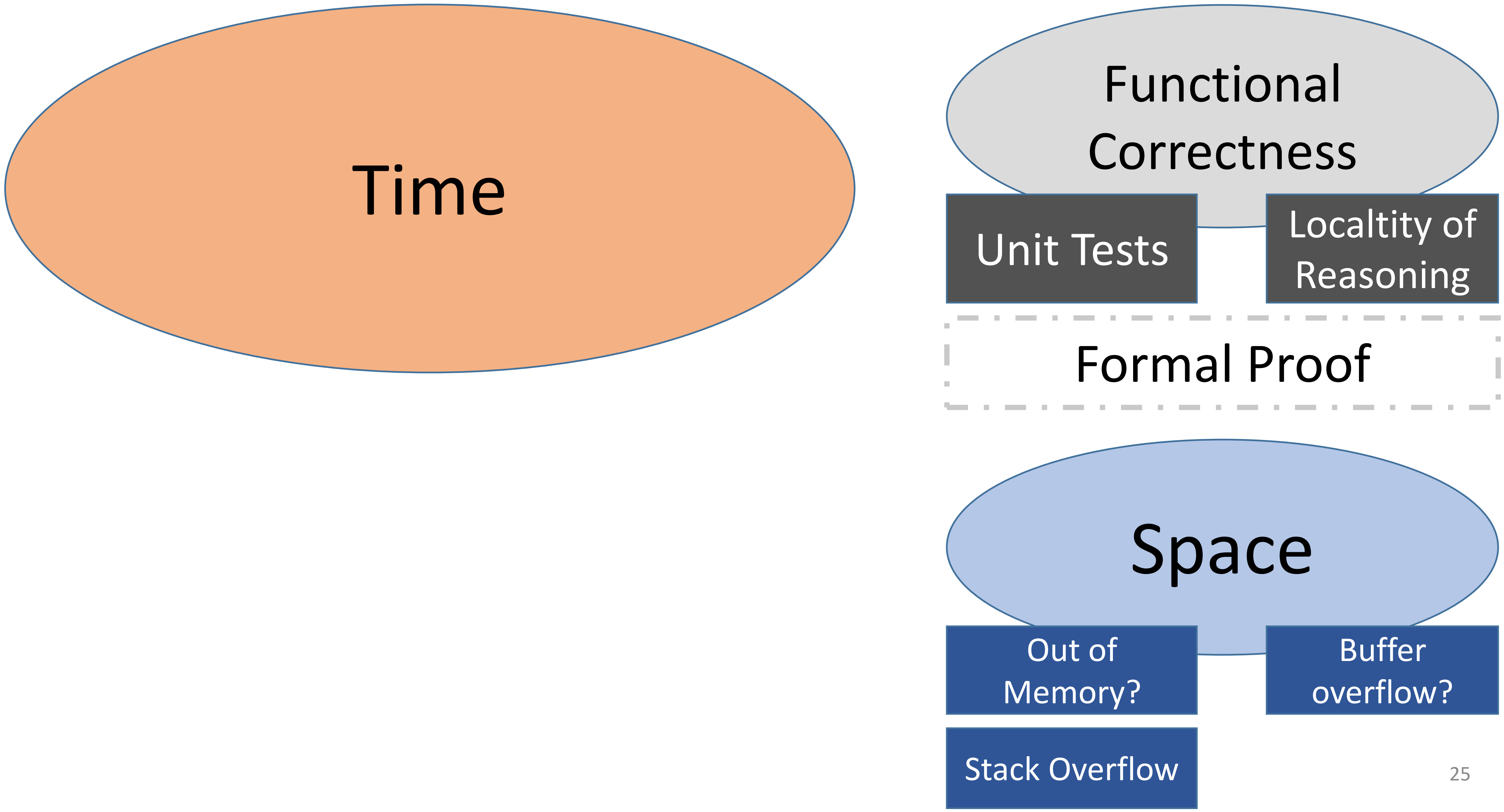

Functional Correctness
Space
Tests
Localtity of Reasoning
Formal Proof
Out of Memory?

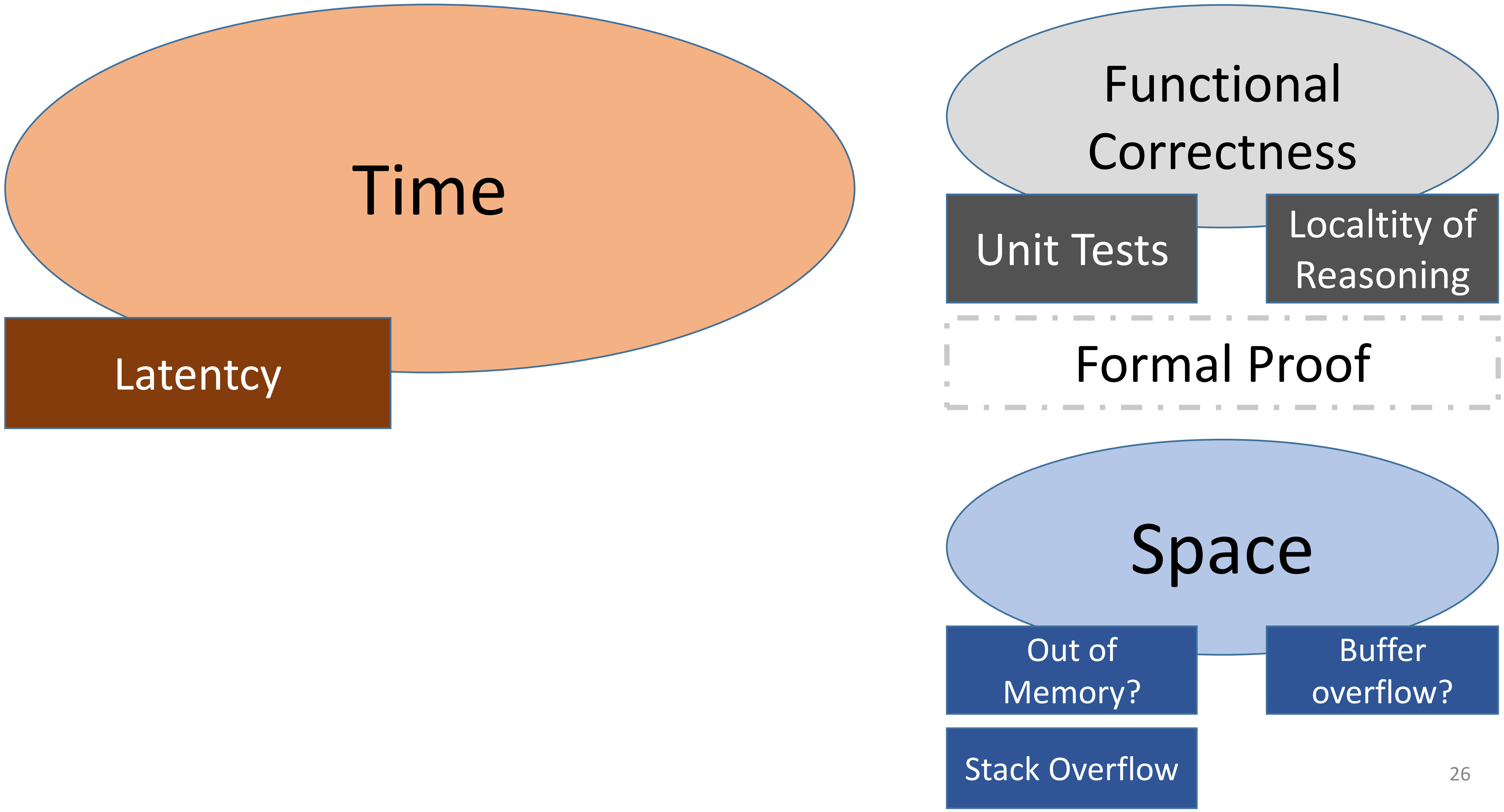

Functional Correctness
Space
Tests
Localtity of Reasoning
Formal Proof
Out of Memory?
Buffer overflow?

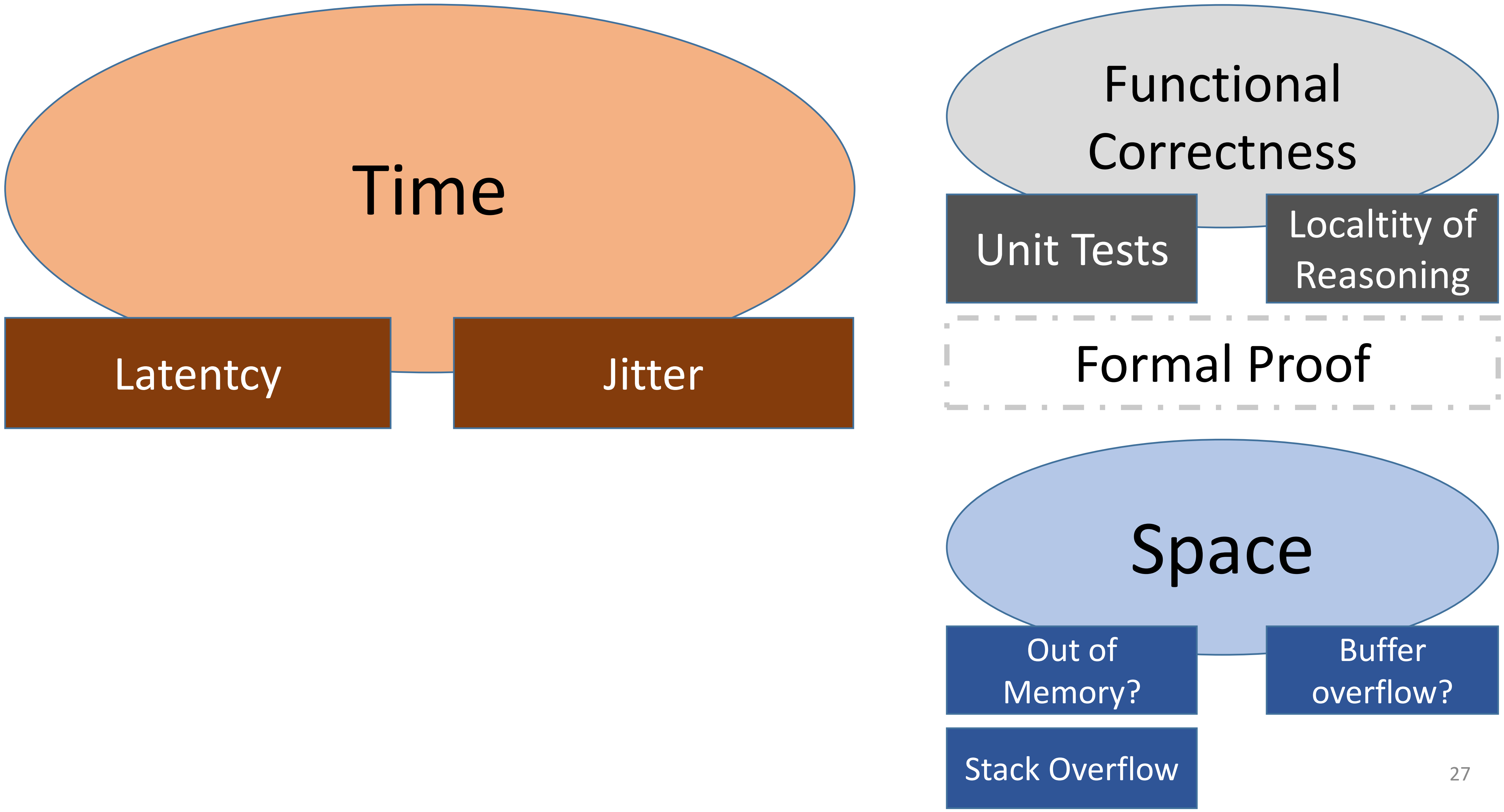

Functional
Correctness
Tests
Localtity of
Reasoning
Formal Proof
Space
Out of Memory?
Buffer overflow?
Stack Overflow

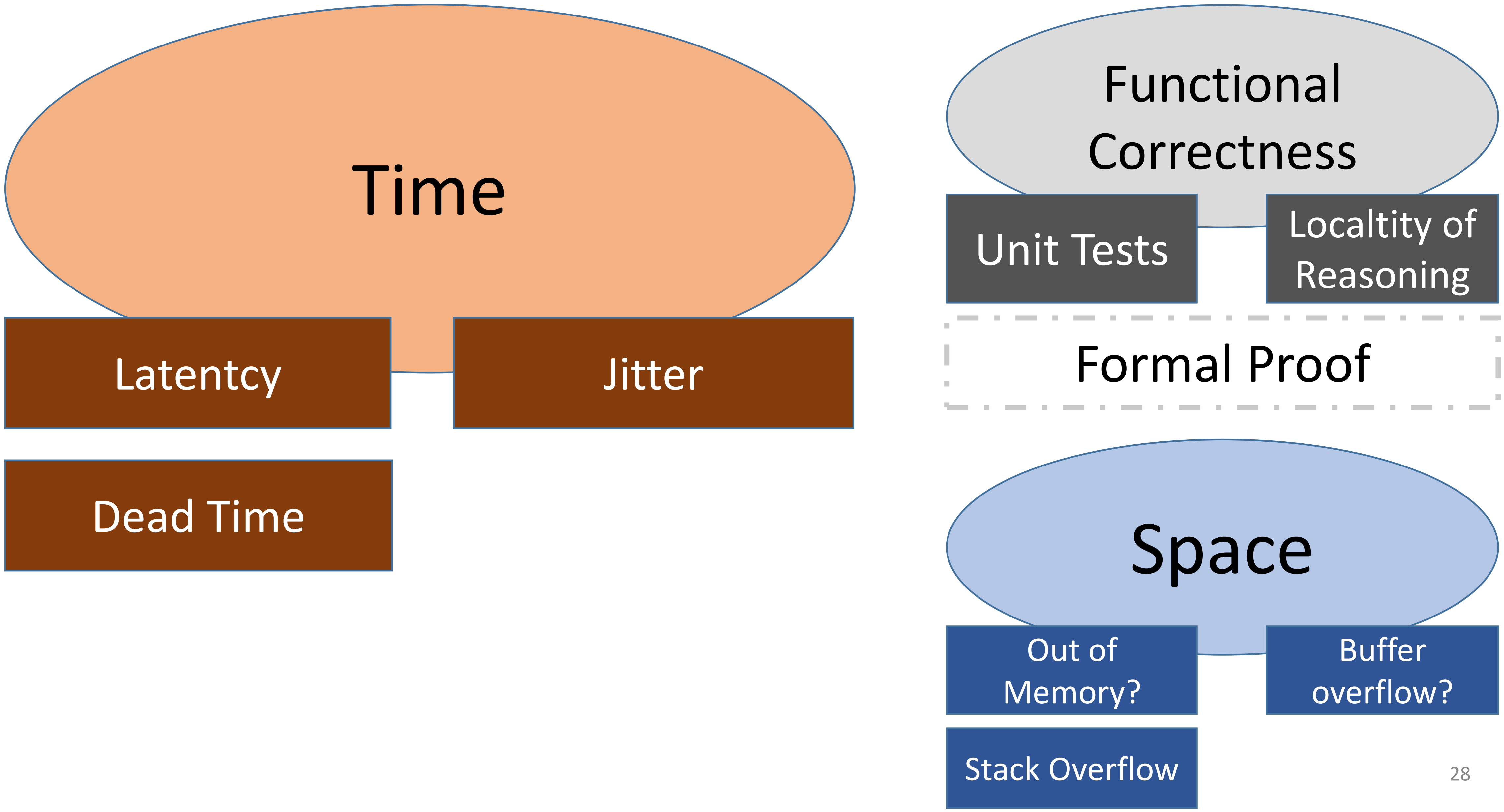
@odinthenerd

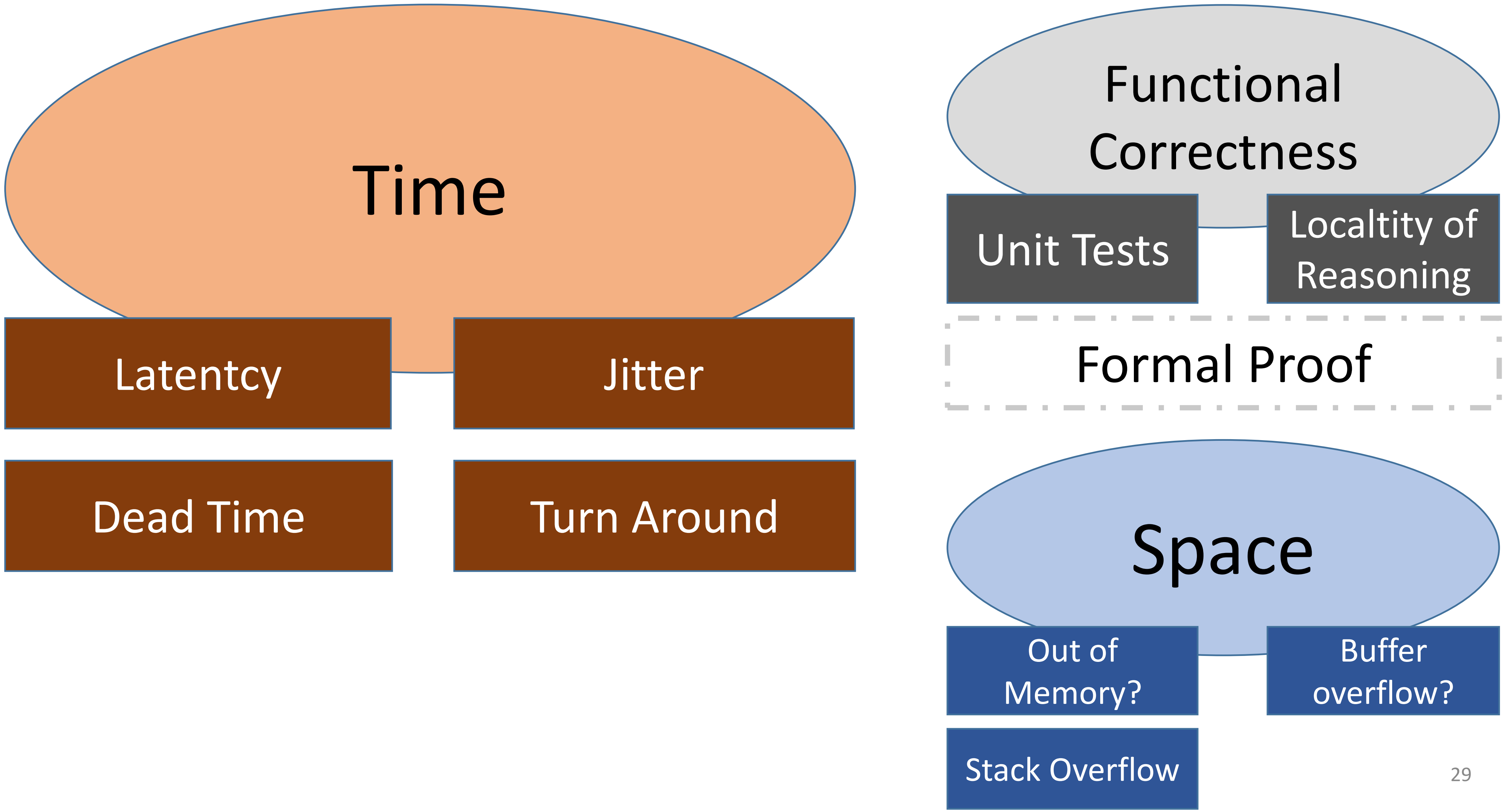


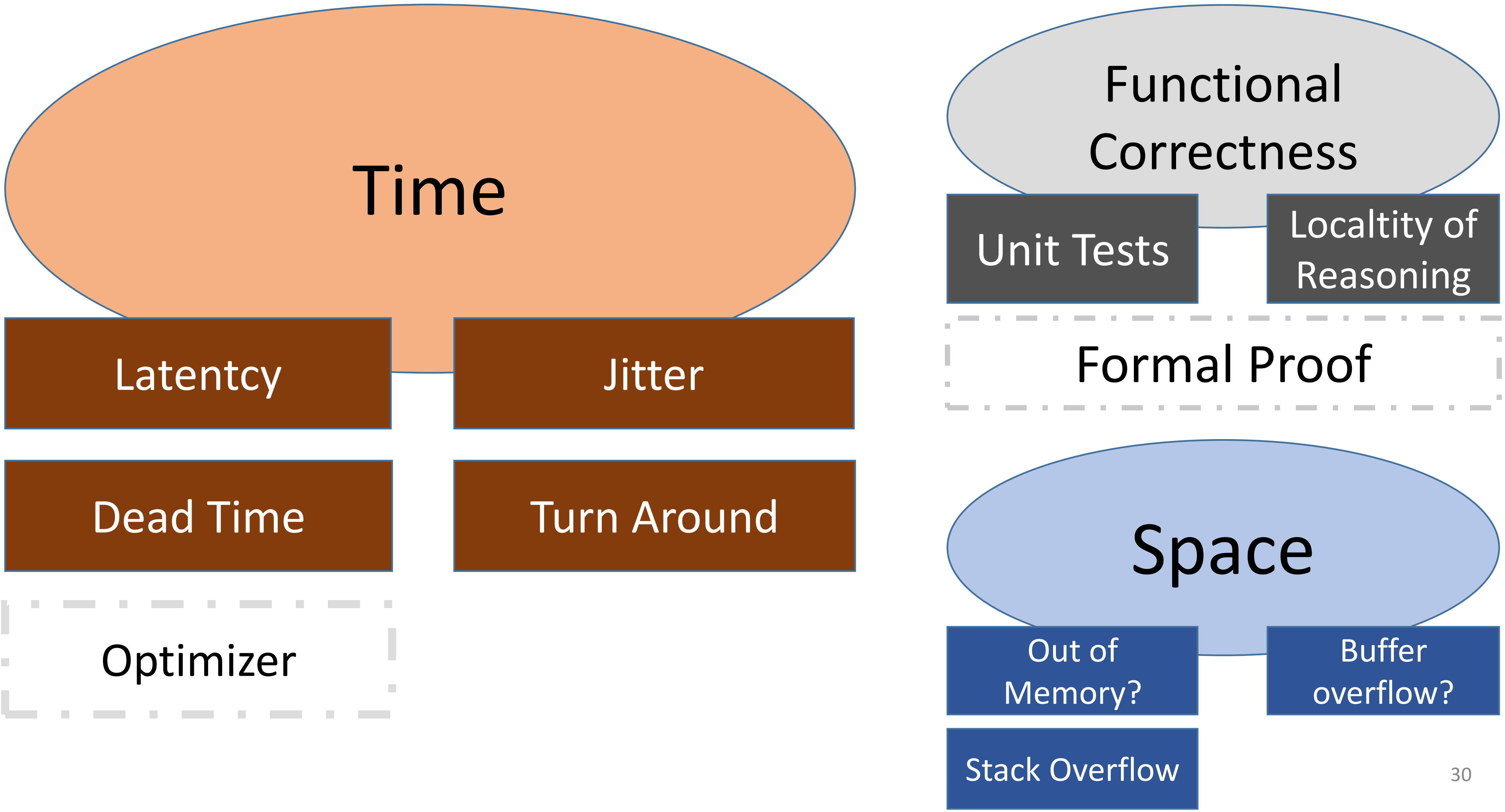

Time
Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof
Space
Out of Memory?
Buffer overflow?
Stack Overflow

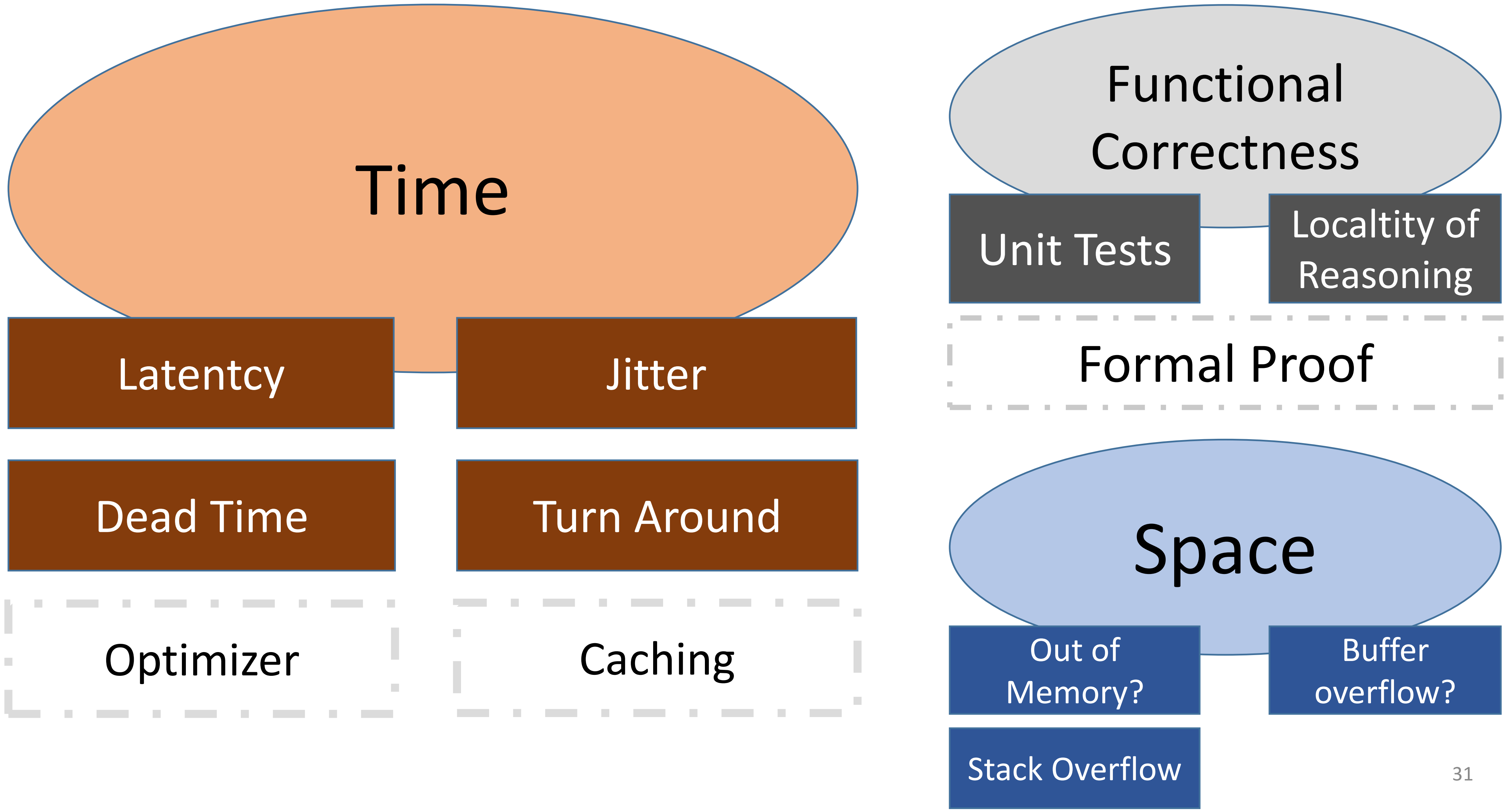

Time
Latentcy
Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof
Space
Out of Memory?
Buffer overflow?
Stack Overflow


Time
Latentcy
Jitter
Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof
Space
Out of Memory?
Buffer overflow?
Stack Overflow


Time
Latentcy
Jitter
Dead Time
Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof
Space
Out of Memory?
Buffer overflow?
Stack Overflow


Time
Latentcy
Jitter
Dead Time
Turn Around
Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof
Space
Out of Memory?
Buffer overflow?
Stack Overflow


Time
Latentcy
Jitter
Dead Time
Turn Around
Optimizer
Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof
Space
Out of Memory?
Buffer overflow?
Stack Overflow


Time
Latentcy
Jitter
Dead Time
Turn Around
Optimizer
Caching
Functional Correctness
Unit Tests
Localtity of Reasoning
Formal Proof
Space
Out of Memory?
Buffer overflow?
Stack Overflow

@odinthenerd

Race conditions

```
void some_ISR(){
    global_thing++;
}

int main(){
    --global_thing;
    return 0;
}
```

Race conditions

```
auto w = global_thing;




--w;
global_thing = w;
```

Race conditions

```
auto w = global_thing;

    void some_ISR(){
        global_thing++;
    }

--w;
global_thing = w;
```

Atomics to the rescue?

```
std::atomic<int> global_thing;
void some_ISR(){
    global_thing++;
}

int main(){
    --global_thing;
    return 0;
}
```

Atomics to the rescue?

```
std::atomic<int> global_thing;
void some_ISR(){
    global_thing++;
}

int main(){
    --global_thing;
    return 0;
}
```

sorry, unimplemented: Thumb-1 hard-float VFP ABI

Emil Fresk

What abstractions can I use?

What abstractions can I use?

Register abstraction DSL

```
struct my_thing{
    static constexpr auto init = list(
        set(thing_hw::super_bit),
        clear(thing_hw::naughty_bit),
        unsafe_write(42_c,thing_hw::danger_zone));
};
```

Chiel Douwes

Au

What does the standard library provide?

- **`<initializer_list> <cstddef>`**
- **`<cstdarg> <type_traits>`**
- **`<cstdint> <climits> <atomic>`**
- **`<new> <limits>`**
- **`<typeinfo> <exception> <cfloat>`**
- **`<cstdlib>`** (program startup and termination)
- **`<ciso646> <cstdalign> <cstdbool>`**

How to target 10,000 platforms

-
-
-

How to target 10,000 platforms

- use an internal code generator
-
-

How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
-

drivers

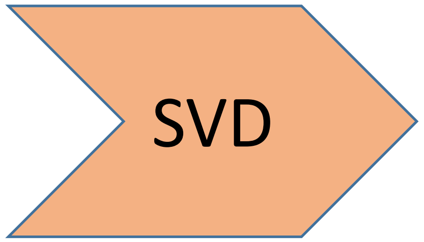
```
auto myUart = make_uart(
        interface<blocking_tx>,
        uart1,
        9600_baud,
        rx = 0.9_pin,
        tx = 0.10_pin
);

myuart.blocking_send("hello world");
```

How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan

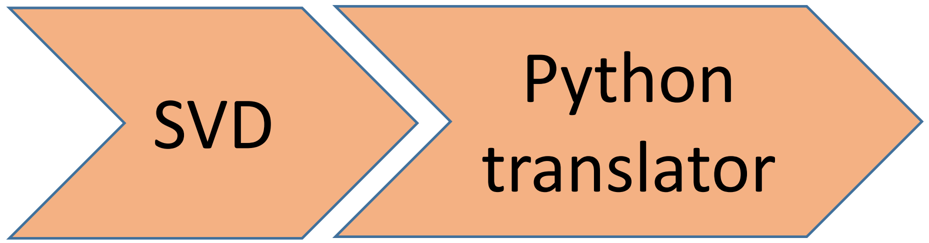
How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan



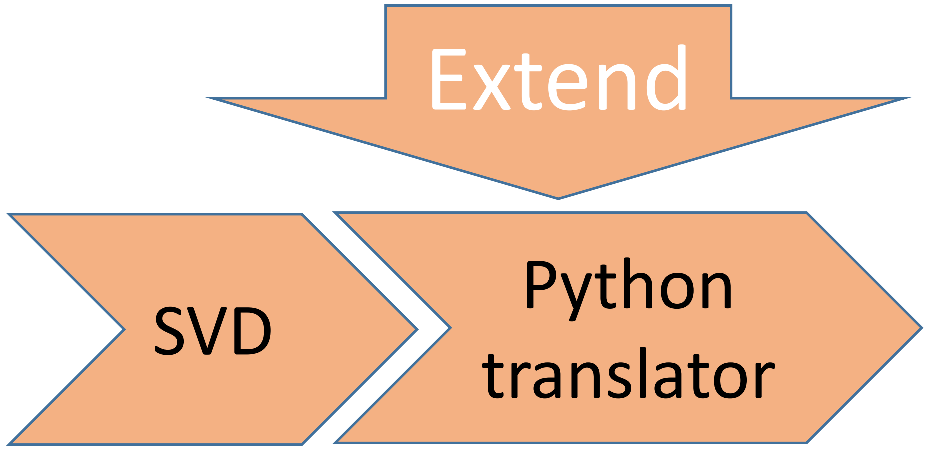
How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan



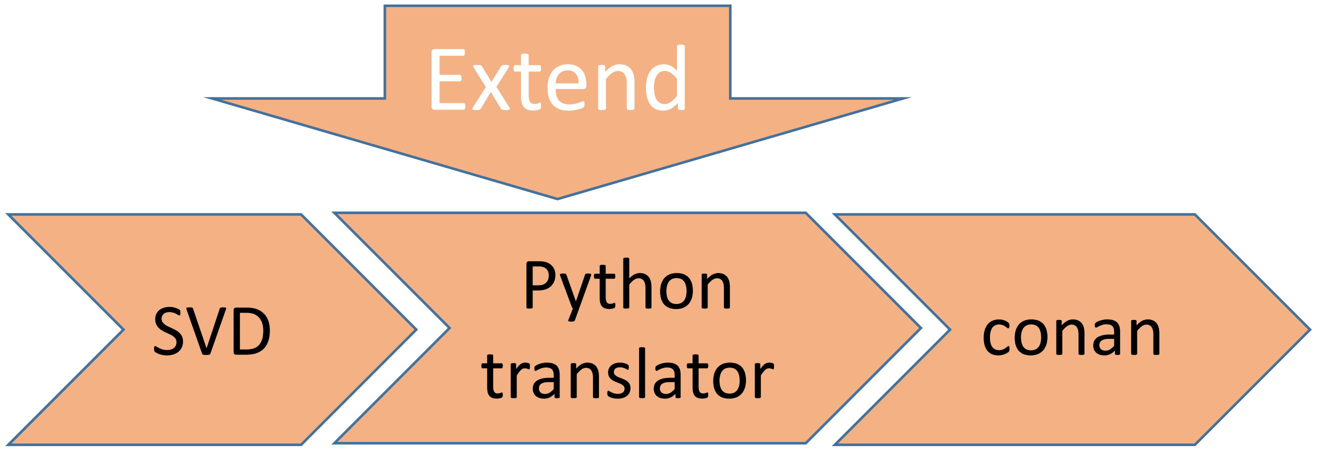
How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan



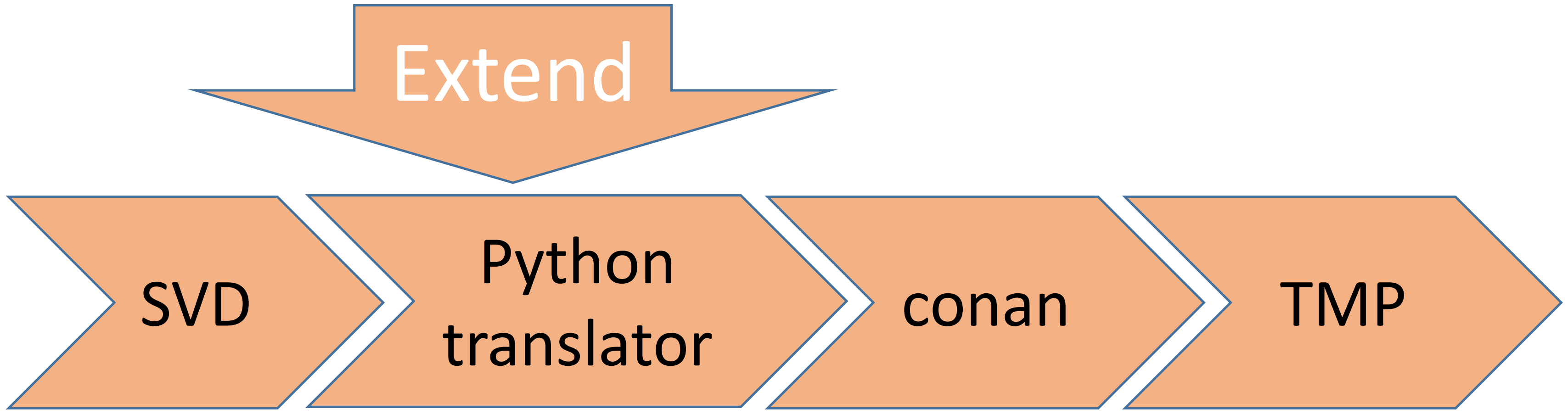
How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan



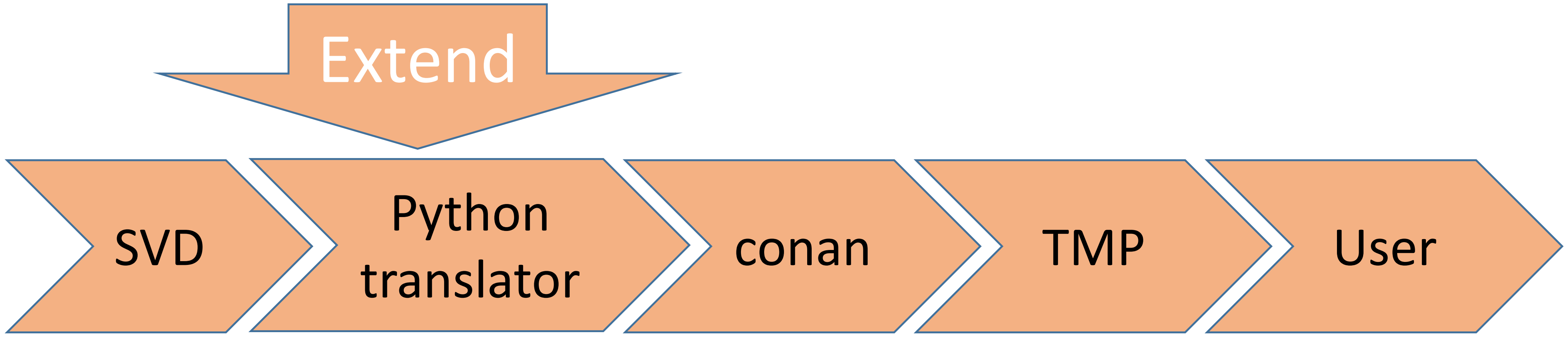
How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan



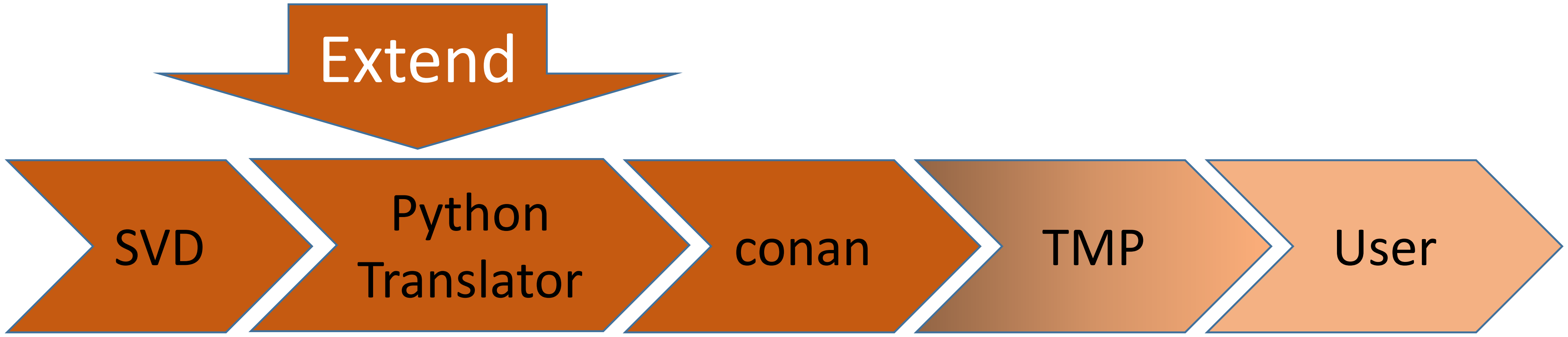
How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan



How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan

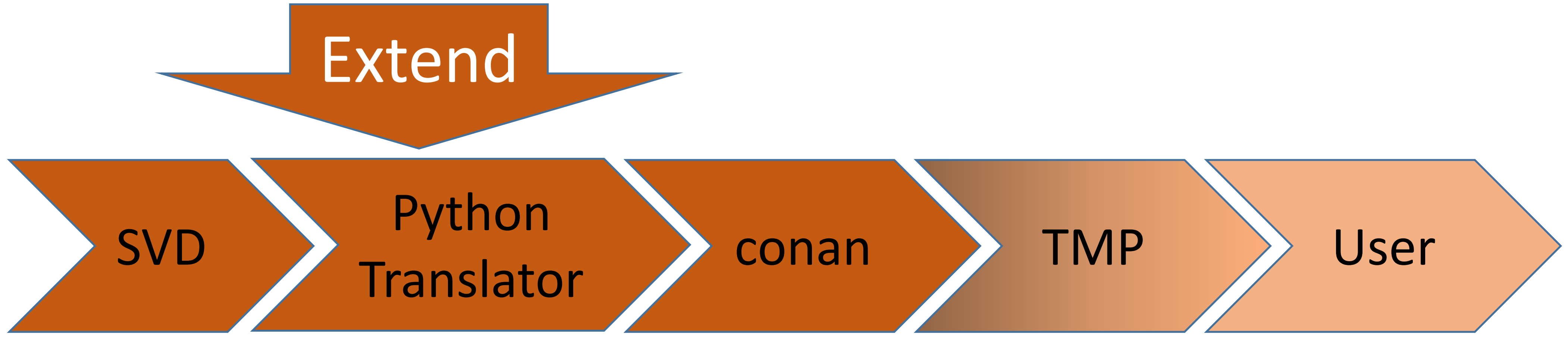


drivers

```
auto myUart = make_uart(
        interface<blocking_tx>,
        uart1,
        9600_baud,
        rx = 0.9_pin,
        tx = 0.10_pin
);

myuart.blocking_send("hello world");
```

How to target 10,000 platforms

- use an internal code generator
- use a declarative paradigm
- use Conan



Event based paradigm

- Sleeping is trivial
-
-
-
-

Event based paradigm

- Sleeping is trivial
- High priority latency tool enforceable
-
-
-

Event based paradigm

- Sleeping is trivial
- High priority latency tool enforceable
- Stack depth low and enforceable
-
-

Event based paradigm

- Sleeping is trivial
- High priority latency tool enforceable
- Stack depth low and enforceable
- Traditionally event handlers = hard
-

Event based paradigm

- Sleeping is trivial
- High priority latency tool enforceable
- Stack depth low and enforceable
- Traditionally event handlers = hard
- Coroutines = awesome

Event based paradigm

- Sleeping is trivial
- High priority latency tool enforceable
- Stack depth low and enforceable
- Traditionally event handlers = hard
- Coroutines = awesome
- We may be able to fix exceptions

The world will be awesome, just different

@odinthenerd

- Github.com
- Twitter.com
- Gmail.com
- Blogspot.com
- LinkedIn.com
- Embo.io