Digital Twins: C++ and Smalltalk

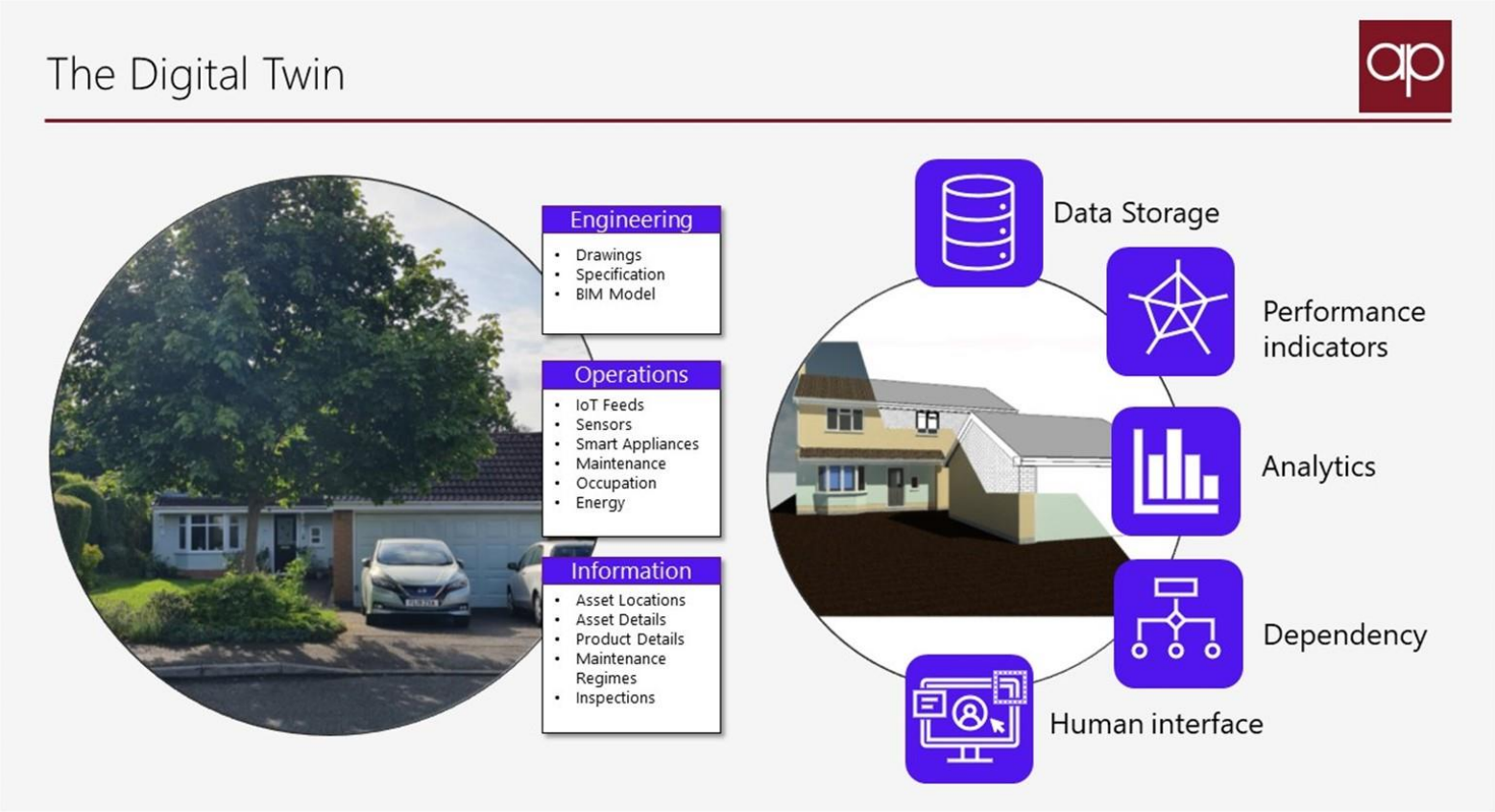
Aik-Siong Koh

2022-09-23 Fri

Motivation

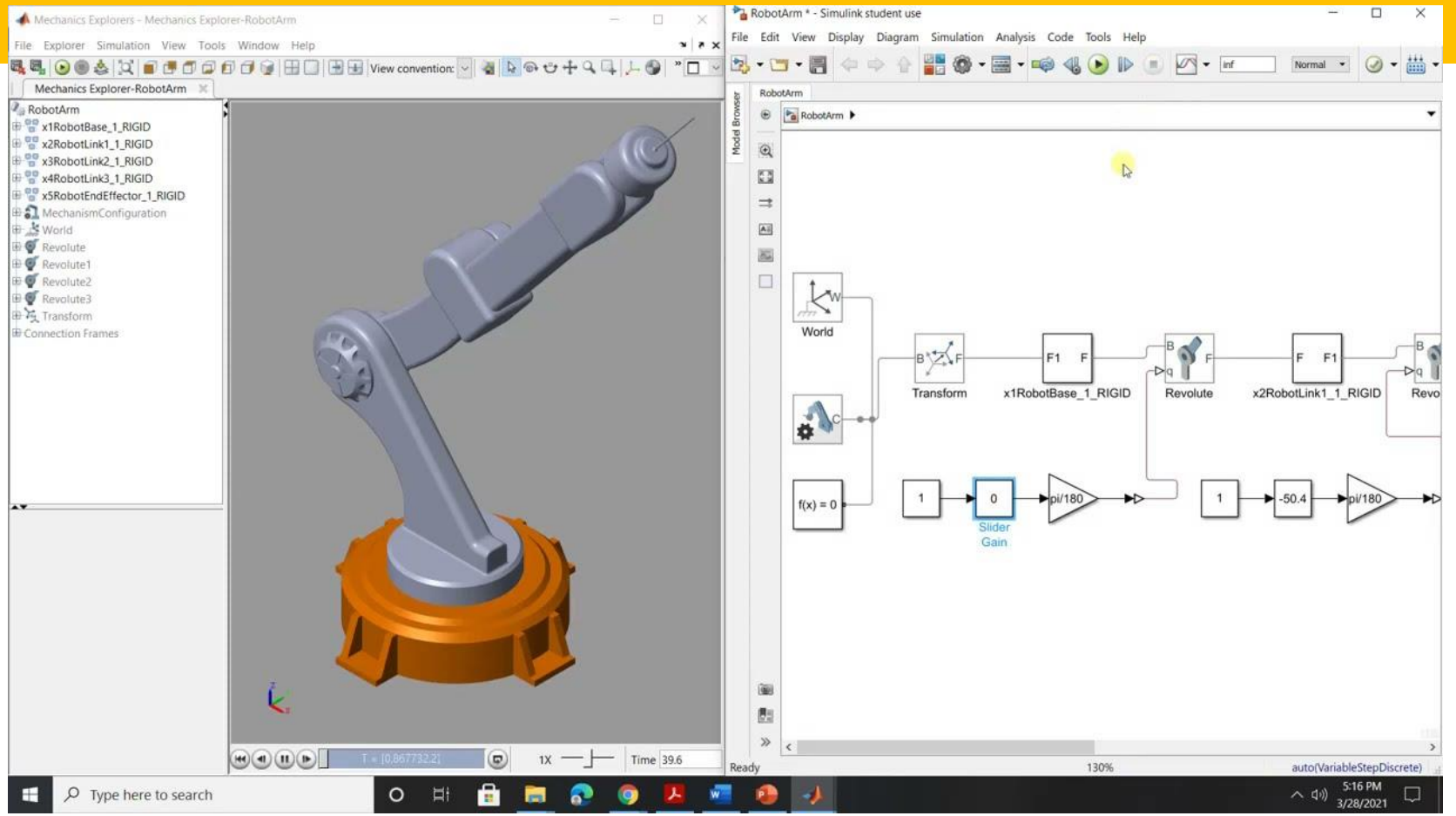
- How to understand complex software better?
- Combine best of static and dynamic languages
  - C++ for speed, Smalltalk for flexibility
- Use redundancy to reduce bugs by orders of magnitude

Digital Twin Concept (2002)



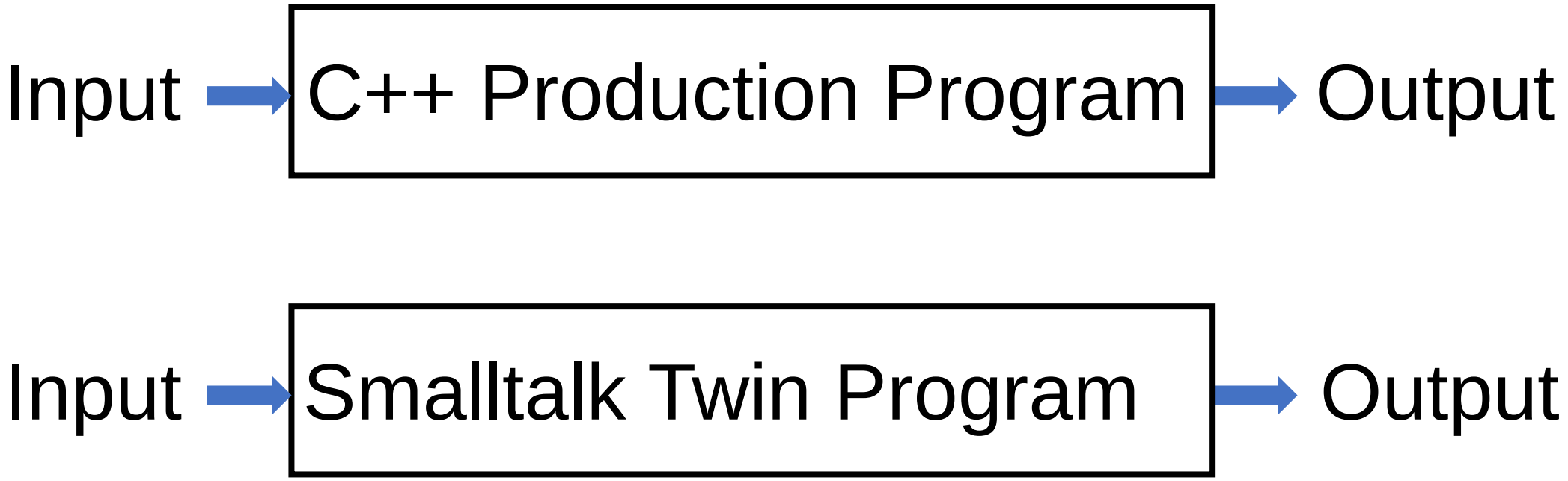
Difficult Easy

Digital Twin Concept



Rigid Flexible

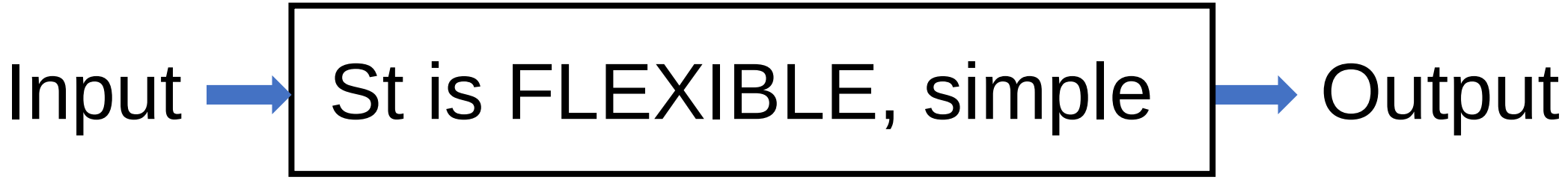
Digital Twins: C++ and Smalltalk



Program can be any size or any component

Why Digital Twins

C++ is best of FAST



Smalltalk is best of FLEXIBLE

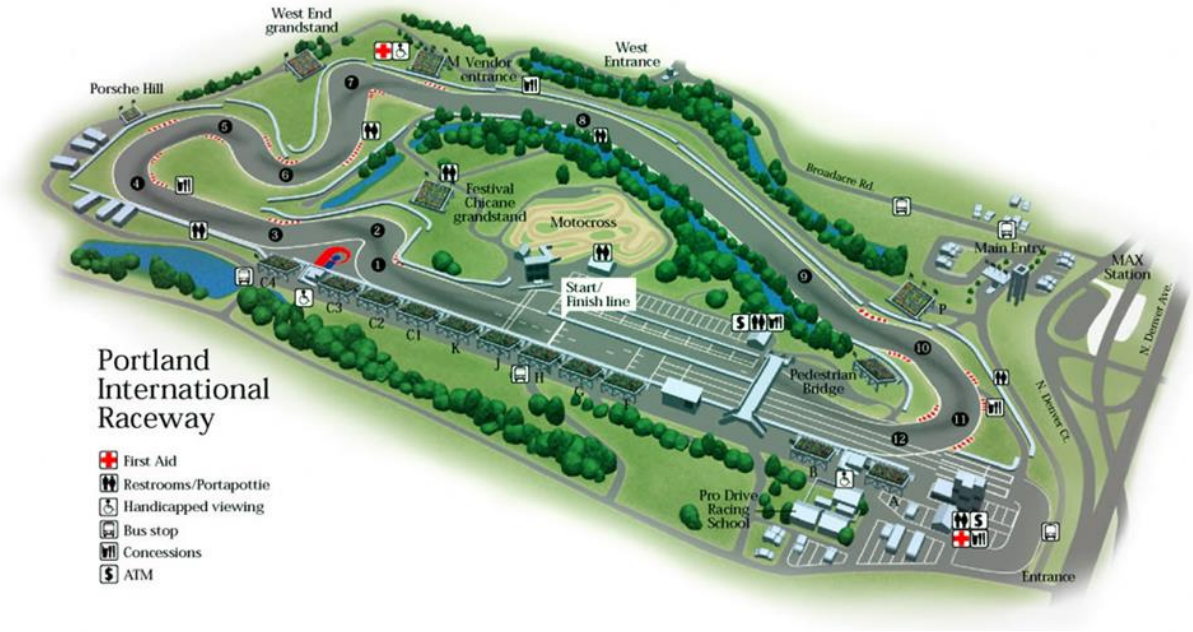
C++ is FAST at all cost

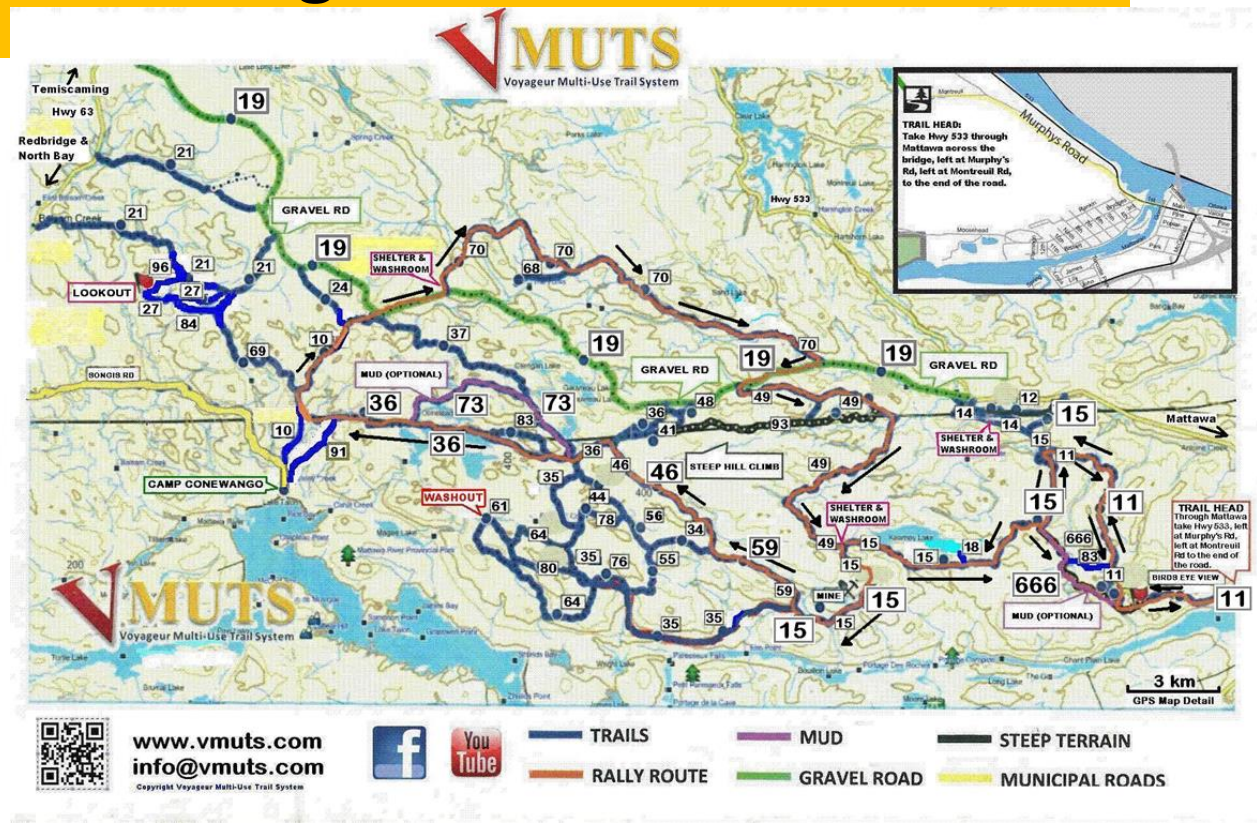
Smalltalk is NIMBLE and rugged
Low cost

A hybrid vehicle would have compromised capabilities
Java, C#, Obj C

C++ Heavy Infrastructure
Small area

Smalltalk Light Infrastructure
Large area

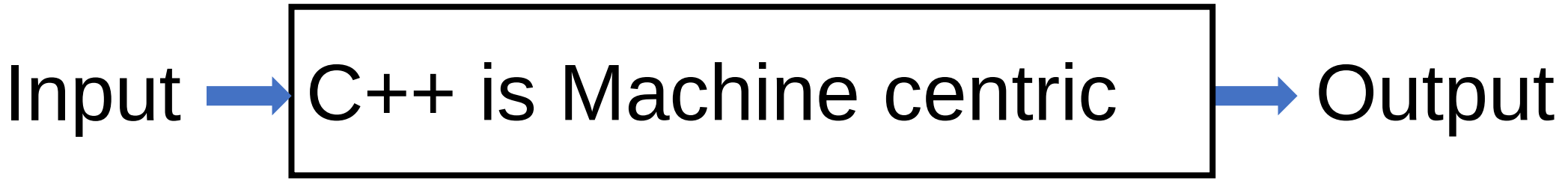




Execution

Exploration

Why Digital Twins 2



Humans think Objects

"Development at the speed of thought"

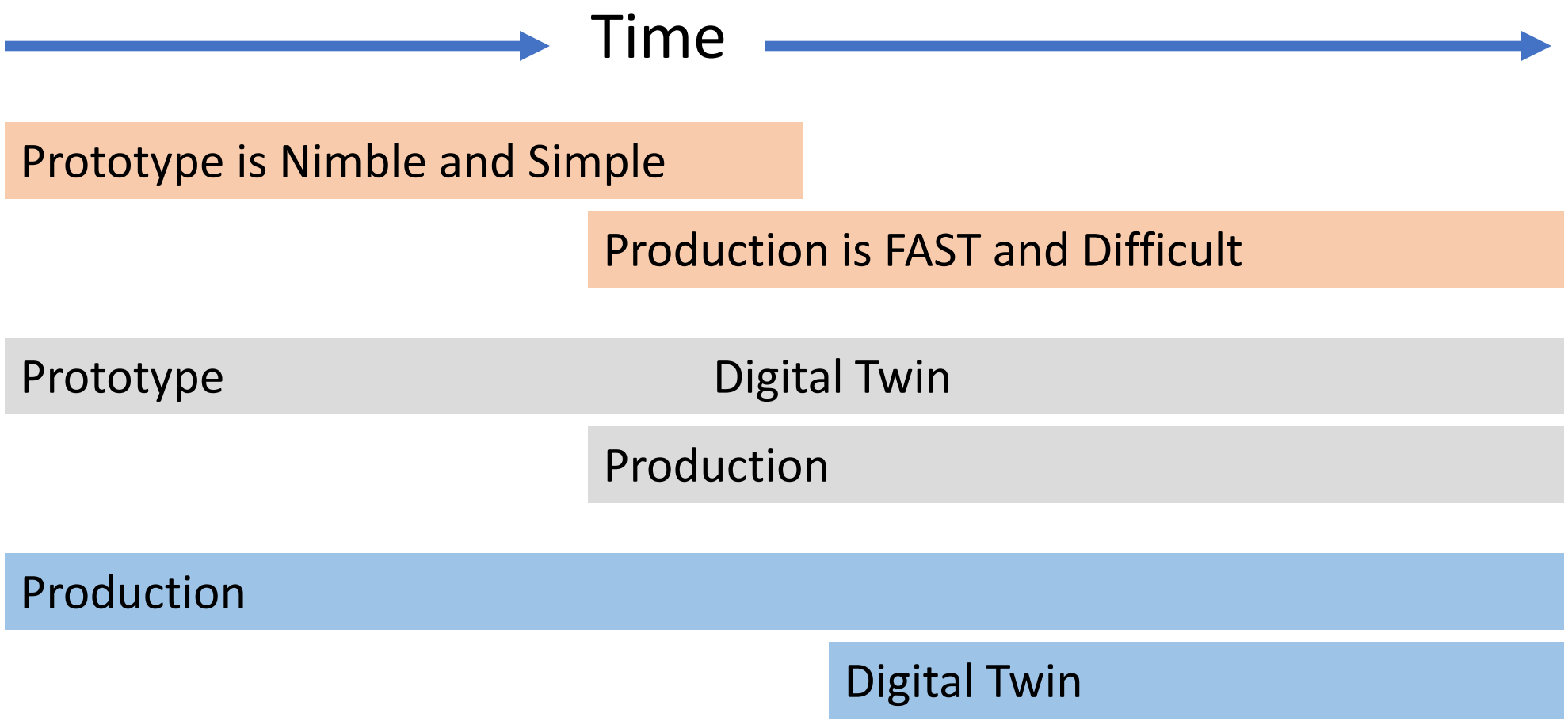
Why Digital Twins 3

- Twin can help documentation
- Twin could be substitute for documentation
  - Smalltalk is readable and less verbose than English
- Twins reduce bugs by checking each other
  - Probability of both making the same bug is product of Probabilities of each

Strategy for Twins

- Capture C++ algorithms in Smalltalk twin
  - Executable documentation
- Experiment in Smalltalk twin (superset)
  - Fearless programming
- Transfer discoveries to C++ twin
  - Manually, automated or both
  - Strict testing
  - Iterate with twin
- Debug in Smalltalk twin
- Transfer fixes to C++ twin

Prototype vs Digital Twin vs Production



Why Smalltalk

- Everything is an object all the time
  - Including IDE, Compiler, Debugger, Scheduler
  - Even when saved to disk on exit
- All operations are through message passing
  - Everything in Smalltalk is written in Smalltalk
    - Including Virtual Machine
  - Critical Smalltalk code are translated to C to be called
    - VM code is translated to C and compiled to executable

Why Smalltalk

- Everything is an object all the time
  - Pure OOP
  - 1, 1.0e2, $a, #b, ‘Hello’
  - true, false, nil, self, super, thisContext
  - Class, Block (lambda), Compiler, Debugger, etc
  - On exit, all objects are saved to disk
  - On startup, IDE returns to exact state before exit
  - Automatic garbage collection

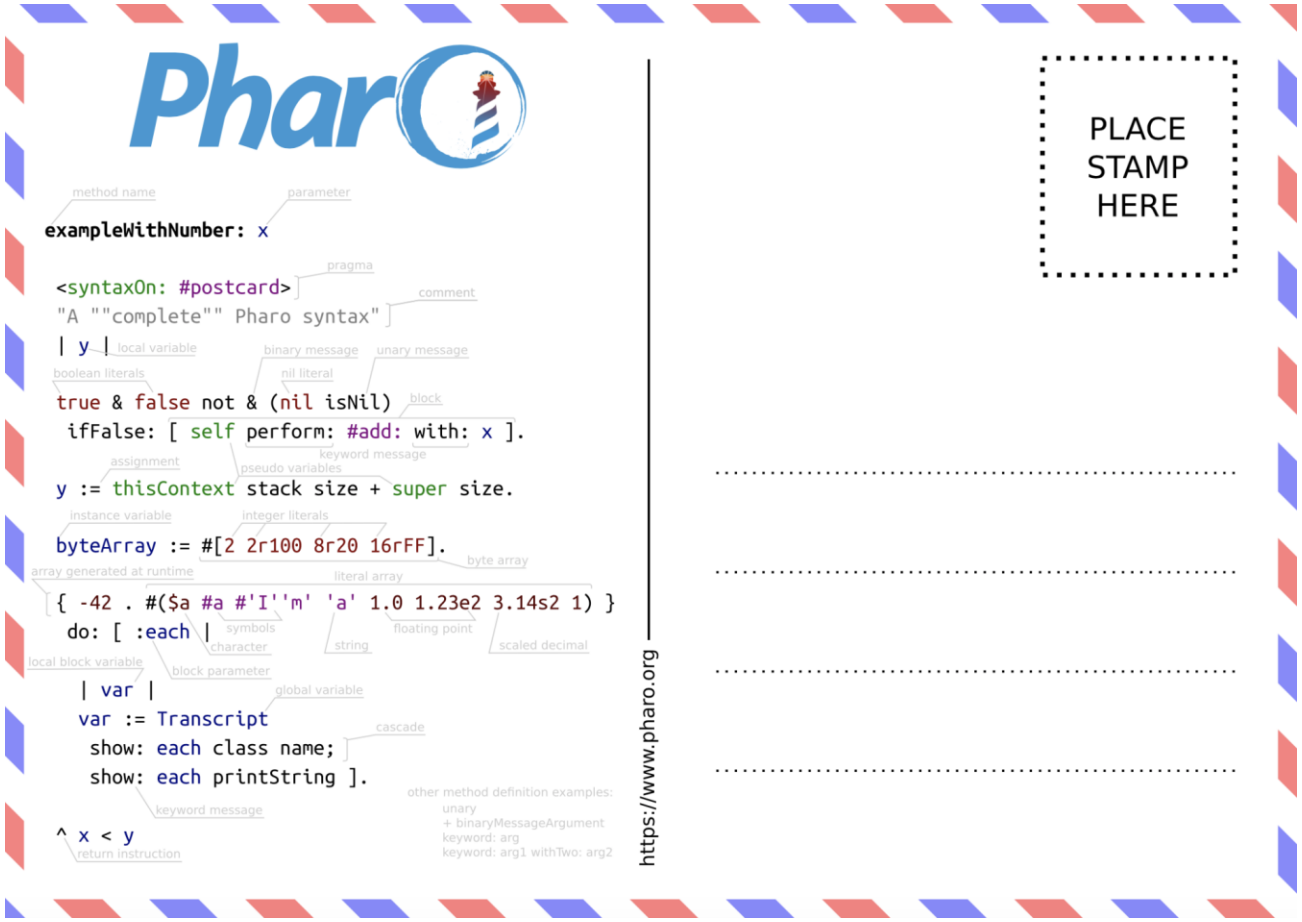
Why Smalltalk

- All operations are through message passing
  - Object message (like Subject predicate)
  - array isEmpty.
  - 4 + 3.
  - Transcript show: ‘Hello World’.
  - array do: [:each | each initialize. each run].

Why Smalltalk

- Smalltalk pioneered Pure OOP
- Influenced Objective-C, Java, Python, Ruby
- Pioneered MVC, GUI
- Xerox invented it for DARPA
- Xerox killed it
  - They wanted to sell printers and copiers
- It is free to be owned

Entire Syntax fits on a postcard



Smalltalk is Best of Flexible

- Edit and run anywhere
  - Zero build time
- Edit and continue in Debugger
- Inspect any object anytime
- Arbitrary levels of inspecting and debugging
- Save all objects and restart where you left off

Digital Twins: C++ and Smalltalk

Hybrid program would be a mediocre compromise

Smalltalk VM is written in Smalltalk and Translated to C

- Slang is subset of Smalltalk
- Slang mimics C
- VM is written in Slang in Smalltalk IDE
- VM is simulated in Smalltalk IDE to run another Smalltalk image
- Full debugging power of Smalltalk used on VM code
- VM Slang code is then translated to C code
- VM C code is compiled to make runtime executable VM

Slang to C translation

Slang (subset of Smalltalk)	C
instanceVariableNames: 'foregroundColor backgroundColor'	sqInt foregroundColor, backgroundColor;
classVariableNames: 'MemorySize'	#define MemorySize 10
^a+b	return (a+b);
a bitShift: 4	a >> 4;
now := FooBar foo: x bar: y	now = foobar(x,y);
^self bigEndian ifTrue: [16r6502] ifFalse: [16r0265]	return bigEndian() ? 0x6502 : 0x0265;
1 to: 10 by: 2 do: [:i | a at: 1 put: (i*2)]	for(i=1; i = 10; i += 2) { a[i] := (i*2); }
flag whileTrue: [self check]	while (flag) { check(); }

getName
  <returnTypeC: 'char *'>
  | newStr |
  <var: #newStr type: 'char*'>
  newStr := 'hello'
  ^newStr
	char *getName(void)
	{
	  char*newStr = "hello";
	  return newStr;
	}

Slang to C translation

Slang	C

defaultWidth
  <inline: true>
  ^10
 	
width
  <inline: false>
  ^width ifNil: [width := defaultWidth].
	static sqInt width(void)
	{
	  return width == nilObj ? (width = 10) : width;
	}

https://wiki.squeak.org/squeak/2267

Smalltalk VM written in JavaScript

- https://caffeine.js.org/beatshifting/
- https://pharojs.org/demo.html

Digital Twins Conclusion

- C++ for speed, strict safety
  - Industry is good at it
- Smalltalk for flexibility, experimentation, fun
  - “Development at the speed of thought”
- Twins reduce bugs by checking each other
- Smalltalk was invented right
- Anyone can “own” Smalltalk for free