YAPS

Yet Another Paradigm Shift

Concurrency and Synchronization

C++ now 2013

stan.lippman@gmail.com

Let Me Introduce Myself

- 1983 :: First introduction to C++ by Jerry Schwarz, implementer of the iostream library, who was designing and implementing NAIL under Steve Johnson in C++ and I was the tester assigned.
- 1985 :: I was the tester assigned to the “brown bag” cfront release 1.0 team.
- 1986 :: I joined the compiler language group under Steve Johnson, and under persuasion from Barbara Moo (bem), worked on creating release 1.1.
- 1986 – midway through release 2.0, I was the only developer responsible for cfront while Bjarne Stroustrup worked on MI.

Let Me Introduce Myself

- I was responsible for the cfront release 3.0 implementation of templates, beginning with a pre-ARM class implementation some folks at an OO database company had developed. That was both a blessing and a curse.
- After that, I left cfront to intern in Area 11, my Lothlorien, in Bjarne's Grail Project. I designed and implemented the Object Model component.
- I was editor of the C++ Report, in the 1990-1996(?) timeframe. I signed on Tom Cargill, Doug Schmidt, Don Box, Scott Meyer, Josee Lajoie, John Vlissides among others to be columnists.
- 2001-2005 :: Architect, Visual C++, but chose to work exclusively on the C++/.NET interface, and worked on the initial redesign of their Managed C++ release. I also wrote a freely available MC++ to C++/CLI translator I call mcfront.

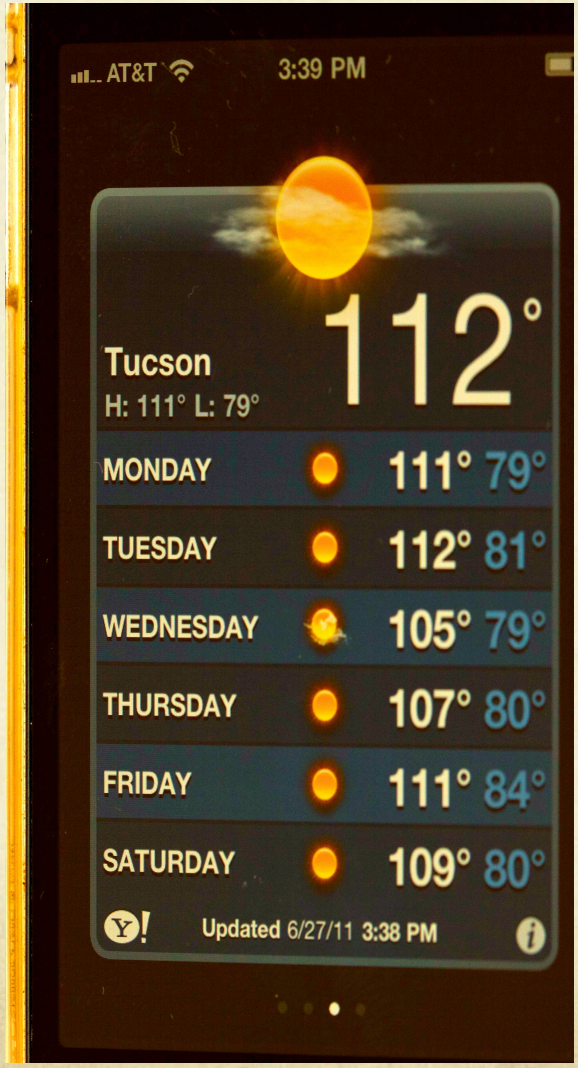
Let Me Introduce Myself

- More importantly, however, when Bjarne's Grail Project was abruptly cancelled and I had three months to figure out where to move my family, I worked in industry, and understood first hand the consequences of the language design choices I had been a part of.
  - Disney Feature Animation, DreamWorks Feature Animation, Pixar Animation, PDI/DreamWorks Animation … [6 or 7 screen credits and lots of t-shirts …
  - Distinquished Consultant at JPL on the use of C++ and OOD, in particular with CLARAty, a freely available modular framework for research.
  - Massive Multiplayer Online (MMO) game companies – Perpetual Technology, which I had signed on to work on the server side of Star Trek, and Emergent Technology, which had the smart idea of writing a Server Framework and a set of high-level scripting languages.

Let Me Introduce Myself

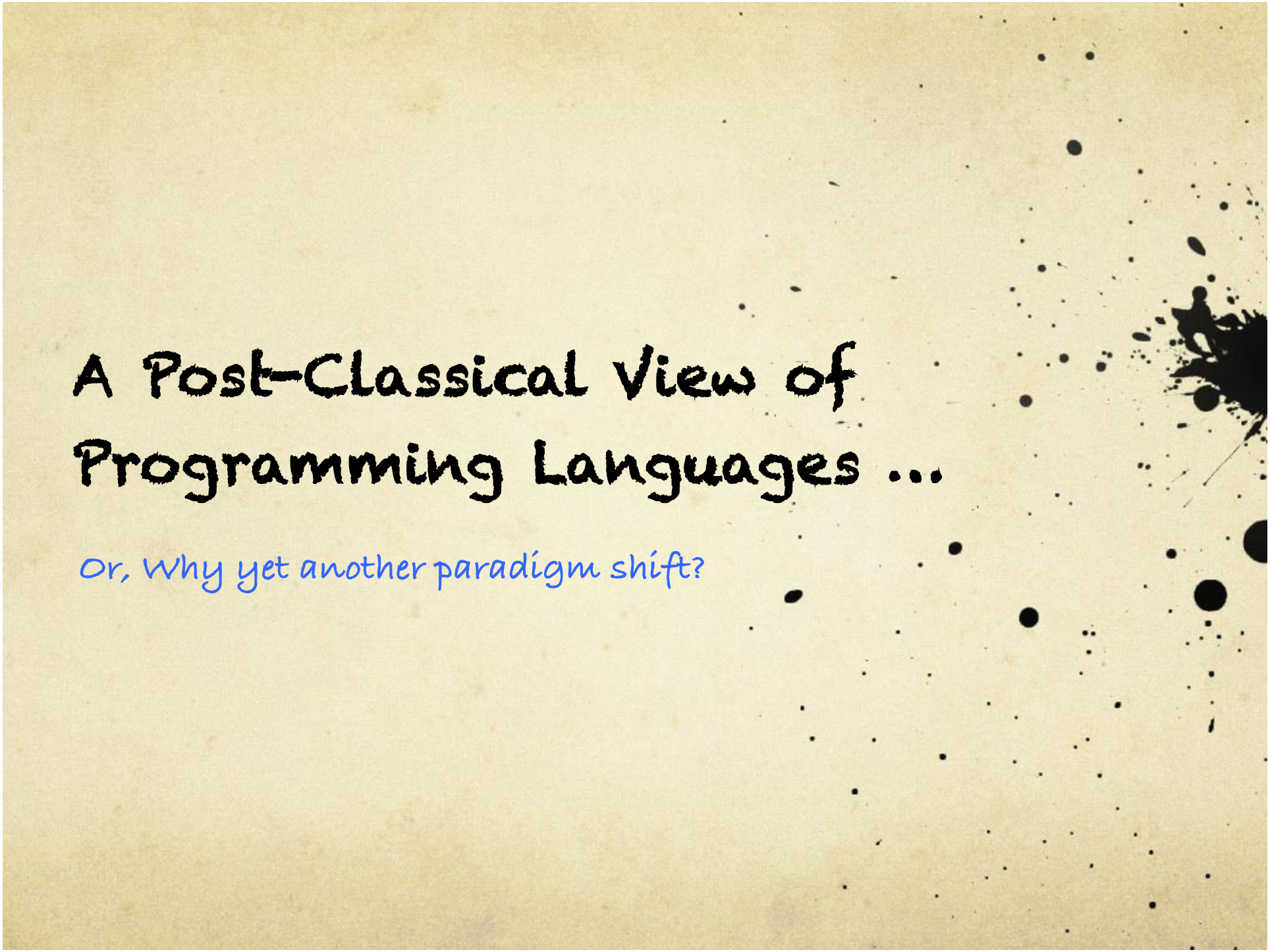
- I have not coded in C++ in almost 5 years, although I have partially coded an Objective-C compiler in Objective-C, and have coded with Objective-C for Apple's iOS – in particular, the iPad.
- I have mostly studied molecular biology and neuroscience, particularly memory and sleep, on my own during this time. I do not read books about coding. I do not make a living by claiming any expertise that people should pay me for revealing to them. That is not what this is.
- I am trying to write an eNovel designing a Narrative Engine and Narrative Science in which to create a Narrative Universe that anyone can add their stories to, following a set of constraints, within a finite but boundless imaginary world housed in the Northwest temperate rain forests of Northern Canada.
- So, this talk for me is just a Sync point (SP) in what has now become now two concurrent but now independent narrative threads: myself and C++ Now.
- Hello ☺

An Intractable Bug ...



A Post-Classical View of Programming Languages ...

Or, Why yet another paradigm shift?



Today's Quiz

Or, Did You Really Think
I Was Going to do All the Work?

What language did the first computer use?

1. A binary machine code?
2. A Hex/Octal machine code?
3. An assembly language?
4. C++ (☺)

You're allowed to rudely shout out an answer …

None of the above!

- Oddly enough, the idea of an independent program – let alone the idea of a programming language – hadn't been part of the original invention of the modern computer.
- Rather, cables were plugged into one configuration for *that* formulation or reconfigured for *this* formulation, and so on.

This is perhaps where the term tight coupling originated … ☺

Programs Dwell in a Computational Environment

- The *modern* computing era *began* without the concept of either a program or a programming language.
- Of course, a transcription of a mathematical formula into a format within the computer was necessary, but
  - it was not thought of as a symbolic program notation
  - there was no concept of inventing a language …
  - there was no such entity thought of as a computer programmer …
- The immediately intractable problems were hardware.

Programs and Program Languages
A Dialectic with the Computational Environment

- The introduction of the program solved *logistical* bottlenecks of the pre-existing computational *environment …*

- Trade-off of decoupling the processing from the program:
  - Faster, `automatic' loading of the program
  - the need to invent and implement a software abstraction layer … in this case, a loader …

This decoupling has been accelerating …

The Evolution of Complex Structure …

- Software did not begin as software – it was a hard-wired configuration – a dance without a separate dance notation.
- This evolved into a reproducible program *bit map* that could be loaded and flushed from memory. A purely numeric representation.
- The assembler formed a nucleus of a symbolic representation of a program but still at the level of individual instructions that could be grouped by function. A mnemonic representation.
- The invention of language to represent a program was both a concession and boon to human nature – it solved another logistical bottleneck.

At each stage, more software complexity is introduced between the program representation and the machine.

A Modern Theory of Programming Languages …

- Programming languages are a response to a particular computational environment:
  - facilitates expression within a current environment …
  - improves on one or a set of existing program solutions …
  - provides a vocabulary and shared point of view …

All Languages Become Extinct …

- As the computational environment changes, the more specialized the language to the previous computational environment, the less adaptive it proves in the new environment …
  - *But the historical accumulation of structure seems to overwhelm these efforts.*
  - *The conditions that give rise to a language leads to its eventual extinction …*

There are many more extinct

than active programming languages

All Languages Compete for Scarce Resources …

- Although a language is not an organism, there is a continual struggle for survival among its population …
  - There is a competition for finite budgetary resources to feed new projects and sustain existing one …
  - There is competition to reproduce in the minds of a new generation of programmers.

  Language wars are virtual bloody both in tooth and claw

All Languages Resist Extinction …

- Typically, the language leaders at some point cease resisting and attempt to readapt the language to the changing environment …
- this however may backfire emphasizing its current maladaptation to the new environment …
  - The population of a language constricts when it fails to reproduce in the minds of the new members of the community
  - Dropping below a certain threshold, it no longer has the critical mass to command finite budgetary resources.

Language as a Unit of Deployment

- **A language is often used as a vehicle for the deployment of a new programming model – that is, of a new paradigm.**
  - **It tends to demonstratively improve on existing models that have run into some bottleneck of scale.**
  - **Or it supports a new model either of technology or abstraction.**
  - **These languages tend to be pure – that is, to provide support for its program model only.**
    - **This makes the language both simpler and more elegant.**
    - **It requires a relinquishment of the past**

In Like a Lion, Out Like a Lamb

- **When there is a reinvention of the dominant program model, there is also a programming language extinction.**
  - **The current generation of languages has no vocabulary to directly express the new model.**
  - **Adding that vocabulary compromises the elegance of the original purity of design.**
  - **A pure language moves from a youthful development community to a acknowledged design influence.**
  - **This passionate sweeping in and hangdog slinking out of programming languages has taken its toll socially on the professional programmer class.**
  - **This is not really working, imo. What kind of solutions suggests themselves?**

Where Can We Go From Here?

- **However, there is a possible language model we can glean from C++.**
  - **What has been *surprisingly* successful for C++ has been its ability to support multiple program models.**
  - **What has been *surprisingly* unsuccessful is the absence of a unifying architecture and crafted boundaries.**
- **Well, perhaps what we need is a conscious design – one that provides quantifiable models of complex systems that can be conserved over time …**
- **Perhaps what we need is an end to the notion of *Great Men*.**

Isomorphic Design Analytics

Getting our heads out of the Clouds that comes of dwelling in a cave ...

The Rational is Imaginary

- Atoms : 1 2 3 4 … infinity (ha!)
- In theory, you just begin with a Positive charge plus a Negative charge, and you get Hydrogen.
- In theory, you just keep adding Positive charge, and the Negative charge will be there.
- None of that is anywhere near the truth. The universe is not Platonic. You cannot reason about it.

Top-down reasoning fails before the material world

Zeno's Paradox

- I am sitting across the room from a door. Rather than get up and walk to it, I think it through first.
- Zeno's Paradox: infinite regress. Result: I don't even try to get up. It is impossible.
- Modern mathematicians prove Zeno's Paradox to be incorrect by exploiting the concept of infinite series.
- None of that is true. It's all in our head. To get to the door, you just have to do it. Thinking fails us because it is not anchored in the material world.

This example is trivial, like our Weather app. But in the everyday world, we fall into this all the time.

Isomorphic Analytics

- How can we think about complexity without falling into Zeno's Paradox? And the antagonisms it engenders?
- Think of temperature before Kelvin. F or C? Both are imaginary. They mean nothing except by convention. Kelvin anchored temperature to the physical world.
- Isomorphic reasoning is an attempt at anchoring design to a quantitative physical model found in nature:
  - The value of the isomorph is quantifiable – we can order isomorphic thinking about a problem without resorting to Authority
  - The isomorph is both invariant across space but mutable over time.

As a first approximation, let's think about Concurrency and Synchronization using the isomorph of iEukaryote fertilization.

Isomorphic Analytics ???

- Natural design is complex, and surprising, but also surprisingly regular, if you enjoy design and implementation issue.
- It exists at four independent but tightly coupled levels that are complex in themselves and yet also built upon what is “below” it.
- It spans the micro scale to the macro scale with all the latency of mechanical constraints and information processing windows in which the information remains relevant.
- That is, there are all sorts of isomorphs just waiting to be applied.
- It is simply a way of rigorously thinking about complex systems that are both concurrent and synchronizing. But I’m not going to argue the point.

Fertilization !!!

- The egg's cell cycle must be suspended, and be able to recognize and restart by a specific event that can occur only during a small window of time that will happen if at all at an indeterminant future time.
- There needs to be a cell mechanism within the egg to bring the second set of chromosomes to a 1-1mapping
- There needs to be an algorithm for recombination to be implemented somehow
- The patterns in this implementation: gradient, receptor, regulatory proteins, a chemical state machine.

Fertilization !!!

- Note that the implementation is typeless – that is, the mechanism to stop and start the cell cycle is more or less invariant, with the code dependent aspects encapsulated in the genome to be developed.
- The mechanism itself is a kind of lisp engine parameterized by the particulate genome or family of genome.
- It is a complete and independent system – the actual delivery system providing the second set of chromosomes – is an implementation detail.
- It has its own physics, time scale, and constraints. It's own vocabulary and domain space. And yet it imposes constraints on everything above it.
- That is, there is not an unconstrained solution space for the delivery system. This is what makes natural design … well, what it is. There is nothing *centrally* human about it because it is bottom-up and conservative.
- In this sense does the mouse, mushroom, and milkweed become one.

Fertilization !!!

- In order to carry out fertilization between animals, bodies emerged – all the existing body plans are said to have emerged during the Cambrian Explosion.
- In any case, bodies developed within a species such that one body form receives the set of chromosomes – let's call it the host.
- And the other body form generates the chromosome set to be delivered – let's call it the packet.
- There is a huge amount of concurrency and syncronization, and even more invention, that needs to be accomplished here!!! Just think about how you would solve all this.

Fertilization !!!

- Need invent a mechanism for projecting cells out of the one body form and being received within the other.
- While this makes the host a reasonably easy delivery target, it is constrained to occur within a window of time in which the two bodies are syncronized.
- Need invent a cell that can be motile, carry its chromosome set, and be equipped with the protein munition necessary to penetrate the egg's membrane.
- Need invent a mechanism to identify an incoming packet as being the right sort of packet. So, better not generate duplicate identities unless you're sure the results won't be catastrophic, even though we're talking millions of years down the line.

Fertilization !!!

- It has to invent an entirely new cell cycle: meiosis …
- For the first time in the history of life, a genome will contain a superset of the genes actually needed within a cell – so a mechanism to manage that has to be worked out.
- For the first time in the history of life, cells will become specialized and form tissue and organ and have a life cycle that extends beyond any one cell. All that has to be worked out.
- This again is an independent level – in this case, all this has to be represented at the level of the genome.
- The genome suddenly must both develop the form and control the function. The earlier regulatory and signaling mechanisms won't scale. An entirely new gen of the genome is required.

So, where are we?

We have looked at two independent levels operating at the microscale, the cellular and the chemical.

Each, while self-contained are tightly coupled. Each operates in a different framework with its own physics and scale of time.

Fertilization !!!

- Now we have crossed over into the macroscale. We have to support the behavior necessary for the species to carry out the host-packet delivery system.
- This is at the level of the nervous system, the run-time OS of the genome, if you will indulge me.
- More seriously, a specie's nervous system is encoded within each individual instantiation of the restarted cell cycle: that is, the union of the set of chromosomes.
- And within that nervous system controlling that form of animal, all the necessary behaviors to carry out the host-packet delivery are over time optimally programmed.
- It is the genome's running program, and within each cell other than the blood cells, a subset of the genome carries out its role in the design and is sensitive to a fixed set of signaling proteins intrinsic to the genome.

Fertilization !!!

- It must develop the motor programs – so the neuron-muscle reflex, rhythmic, and voluntary movements are potentially supported by coming online at different times due to the nature of the nervous system.
- It must develop a syncronization system between the two body forms that triggers the necessary motor programs implementing the host-packet hand-off.
- This requires the invention of modal pathways to receive signals between the two body forms – in mice this is olfactory.
- This requires sensory pathways for analysis (algorithms) and motor response – that is, some sort of representation and look-up mechanism.

Fertilization !!!

- This is where we lose touch with real-time: the pathways are constrained both by the physical transduction, internal representation, and the subsequent processing of that representation.
- This level is where optimization of the form is natural selection at its Darwinian purest: competition among conspecifics within a local neighborhood.
- Literally, this is brain without mind: insects, particularly ants, bees and other eusocial species are the apex.
- Mice have two olfactory systems: a brain-driven appetital and aggressive system between conspecifics, and a more mammalian neural system that allows for environmental smells to enter into the mouse's second nature.

Fertilization !!!

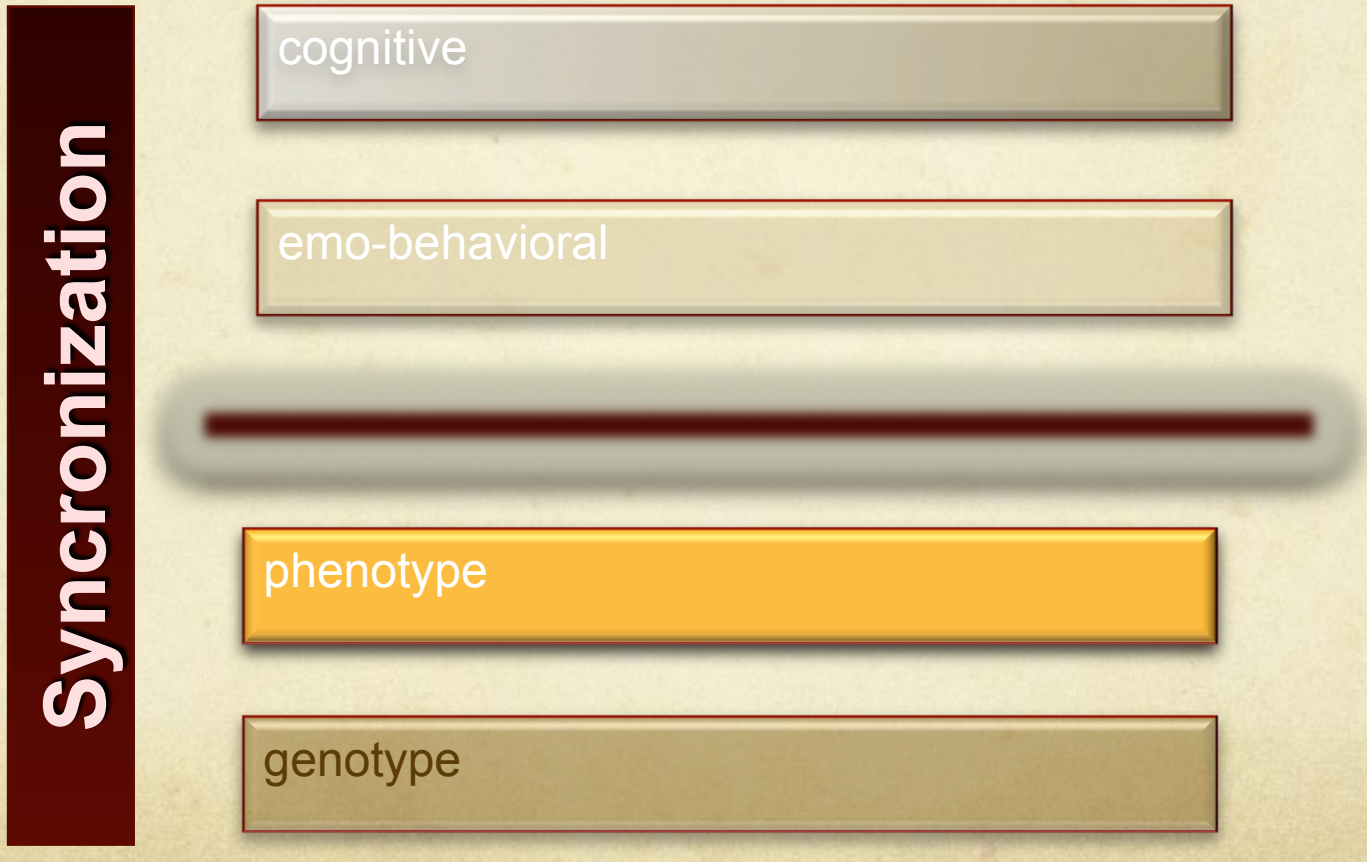
- Finally, in terms of syncronization and concurrency, we have learning and memory, allowing for the emergence of a unifying sentience of varying awareness of self beyond the instance of its embodiment as maintained by its nervous system.
- This is the furthest from real-time, and is both individual to the instantiation and largely private, except as interpreted by its conspecifics.
- So, these two levels are macroscale, and are constrained by the microscale underpinnings. They cannot be directly syncronized because of the boundary nature of Scale.
- But they can be mapped, or a correspondence can be formalized, and can be an object of competition and intelligent design.

So, where are we now?

We have looked at two independent levels operating at the macroscale, the mechanical and the sentient.

Each, while self-contained are tightly coupled. Each operates in a different framework with its own physics and scale of time.

A *meta4tier* Schematic

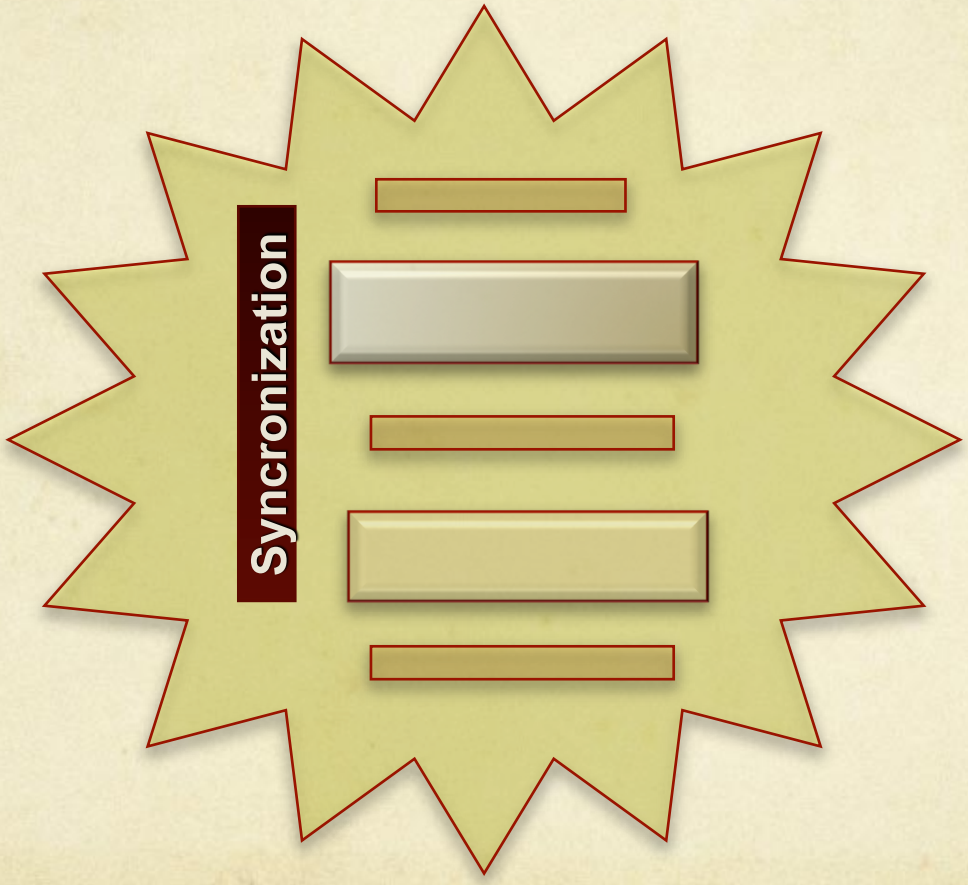


A *meta4tier* Program Schema

Here is the Idea, Idealized ...

- The Schema is Invariant and Fully Specified for the Isomorphic domain.
- It is modular to a specified extent, allowing for swapping out components within the constrains of the overall Schema.
- It is surrounded by software layers that allow development either vertically or horizontally.
- Each layer has its own domain language and domain abstractions, which can be overwritten in the software layers.
- Thus, within the constraints of the schema, the software layers provide multi-tiered abstraction functionality.

Here is the Idea, imaged ...



A Challenge to the Reader

- I'm sure there are a hundred thousand objections to this. Here is my challenge to those objections:
- Design a behaviorial state machine for a mouse-like animal such that
  1. it is responsive both to its environment and conspecifics in a species specific way; but
  2. allowing for learning from experience to individuate behavior.
- This is a complex system. What will be the likely result of this competition, if you will?

A Plethora of Individualized Designs

- I don't doubt every one of you can come up with a smart design and working implementation given the resources and time.
- But I also don't doubt that every one of you will come up with a unique design and implementation.
- Were your professional career trajectory dependent on it you would fight for it tooth and nail.
- It is not possible to achieve a coherent design across the computational space-time.

The Paradigm Shift is an Isomorphism

- All software is obsoleted within a generation, at best. At least everything we've managed to implement so far.
- Let's get off that Merry-Go-Round.
- I understand my meta4tier falls short. It's the best I could come up with.
- So, do something better, then. But not just for yourself, but for a community of minds, if you will. We have a responsibility, as corny as that sounds.
- Let's all work concurrently, and syncronize, and then maybe we can build something that is both complex and coherent across locale and still allow for inventiveness and competition.
- Anyway, this is pretty much what I have to say. If you have any questions, I'll try to answer them. I can't promise that I can. Thank you for listening.

YAPS

Yet Another Paradigm Shift

Concurrency and Synchronization

C++ now 2013

stan.lippman@gmail.com