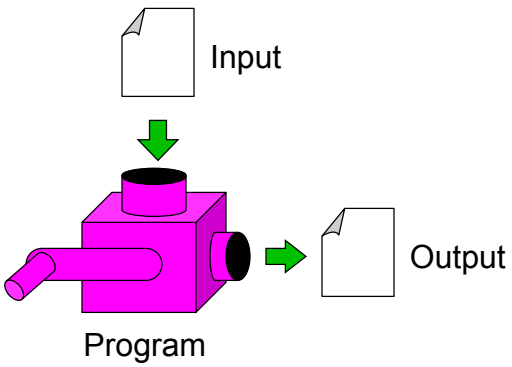
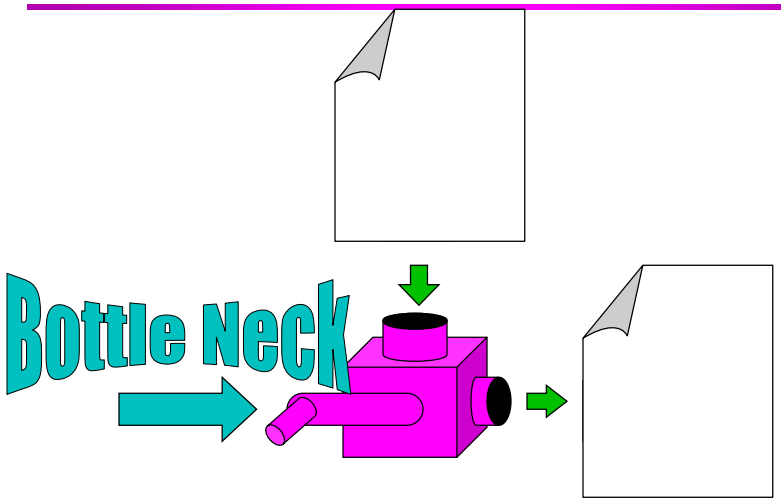
A Pictorial Introduction to Components in Scientific Computing

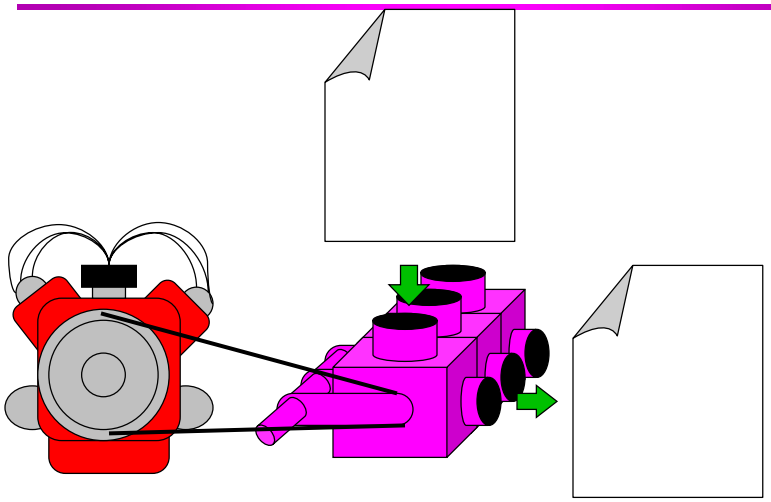
Once upon a time...



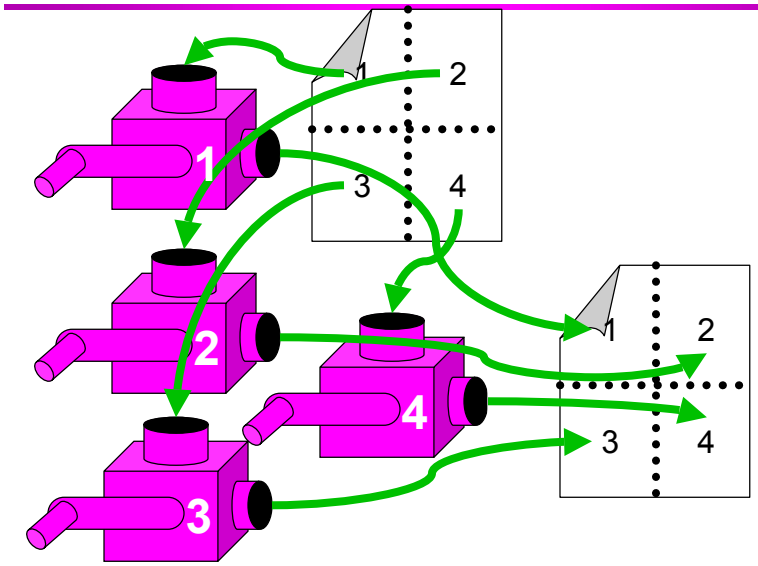
As Scientific Computing grew...



Tried to ease the bottle neck

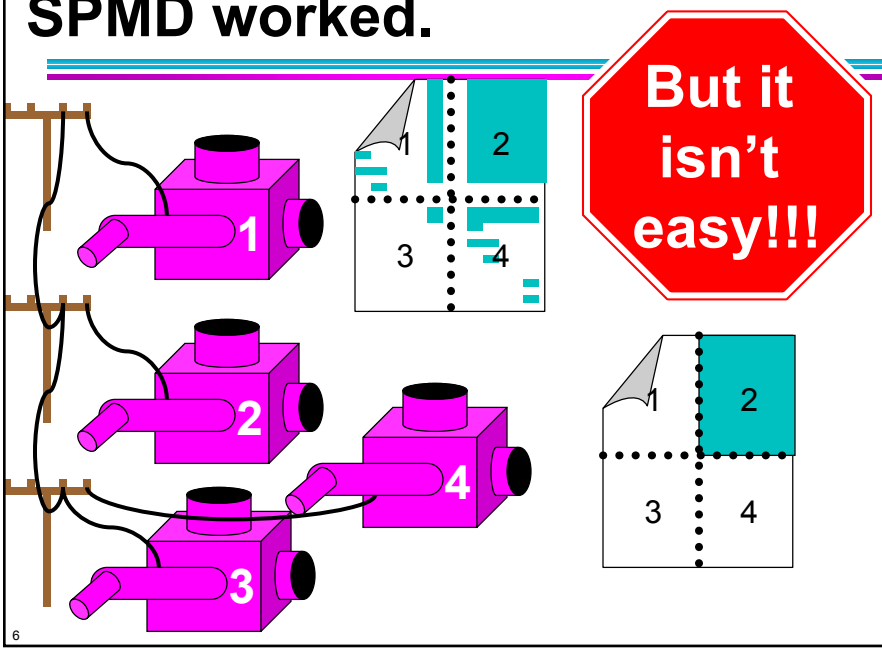


SPMD was born.



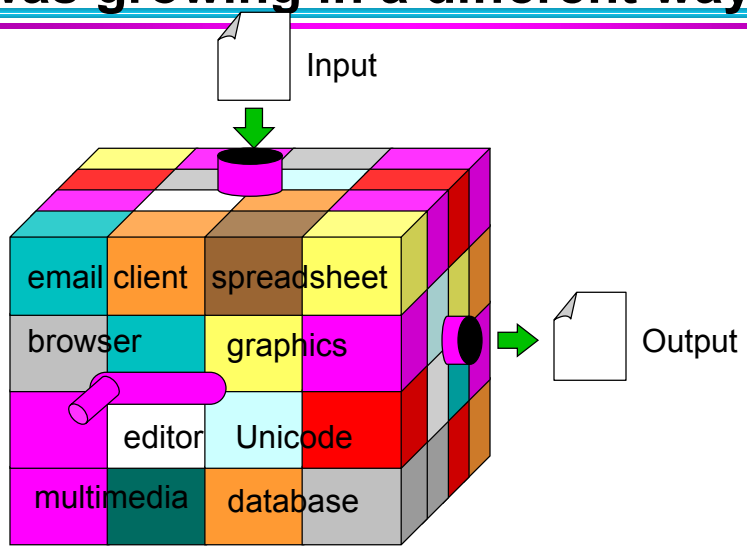
5

SPMD worked.



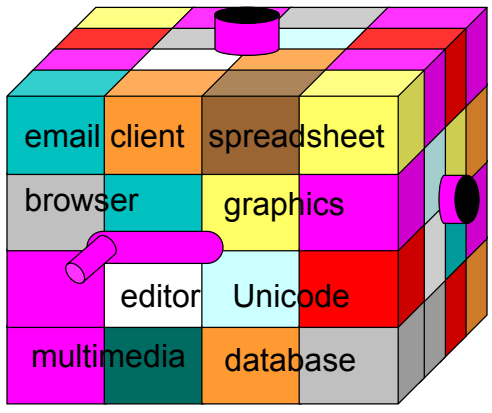
6

Meanwhile, corporate computing was growing in a different way



7

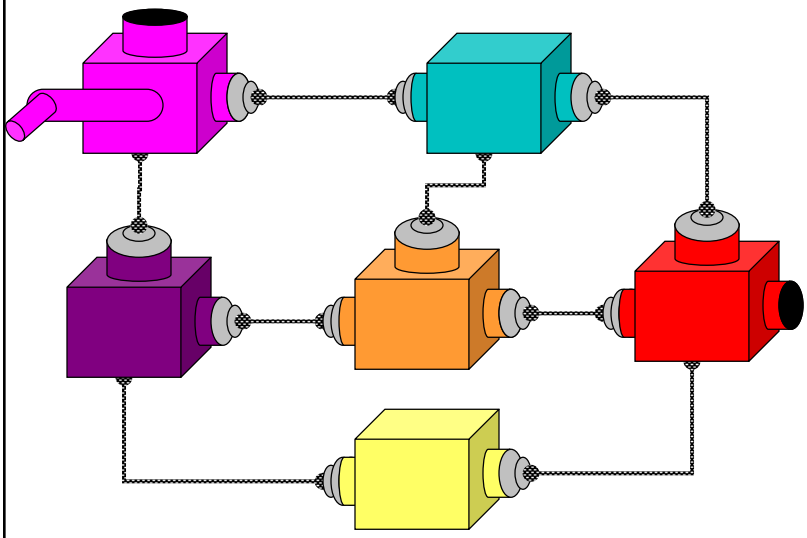
This created a whole new set of problems → complexity



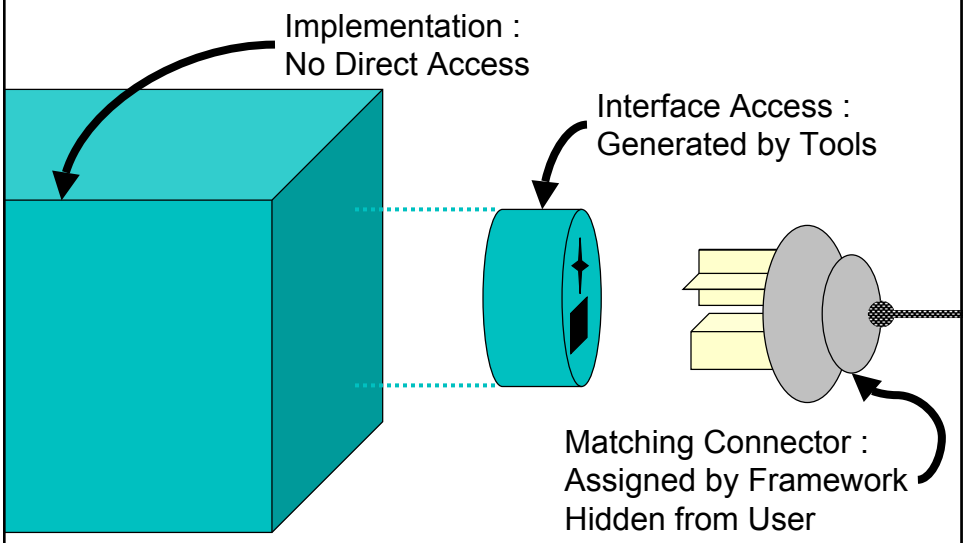
- **Interoperability across multiple languages**
- **Interoperability across multiple platforms**
- **Incremental evolution of large legacy systems (esp. w/ multiple 3rd party software)**

8

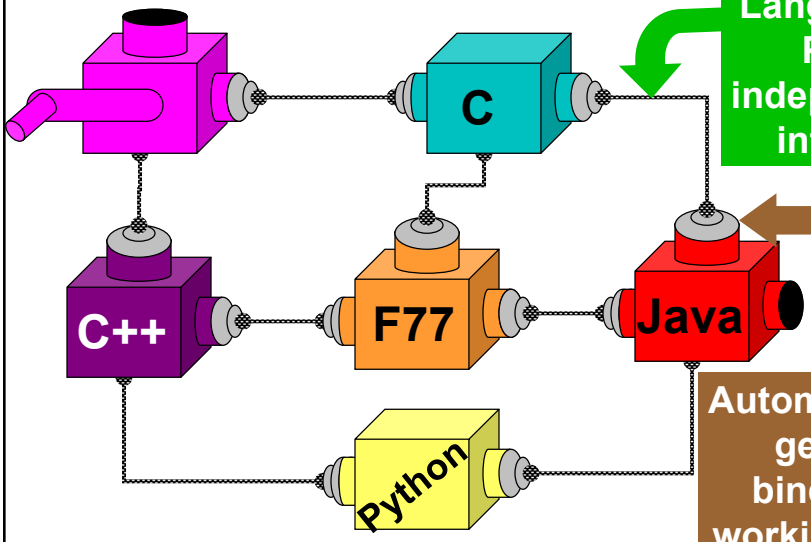
Component Technology addresses these problems



So what's a component ???



1. Interoperability across multiple languages



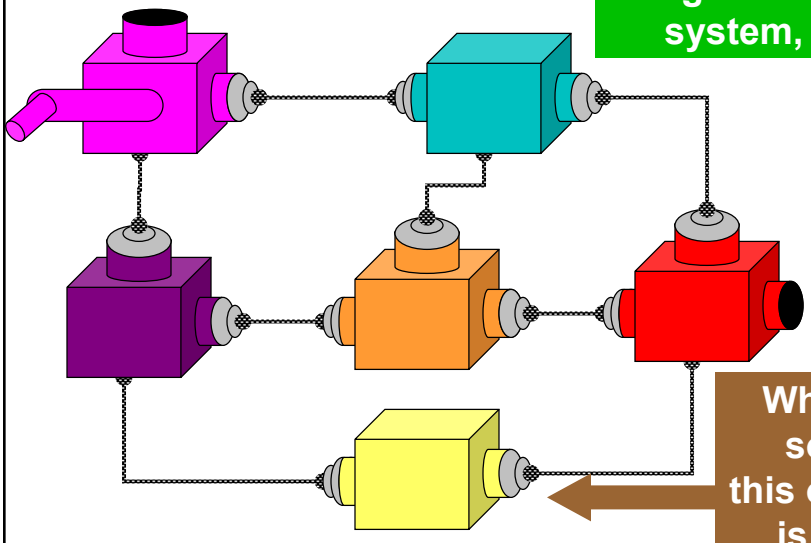
Language & Platform independent interfaces

Automatically generated bindings to working code

11

2. Interoperability Across Multiple Platforms

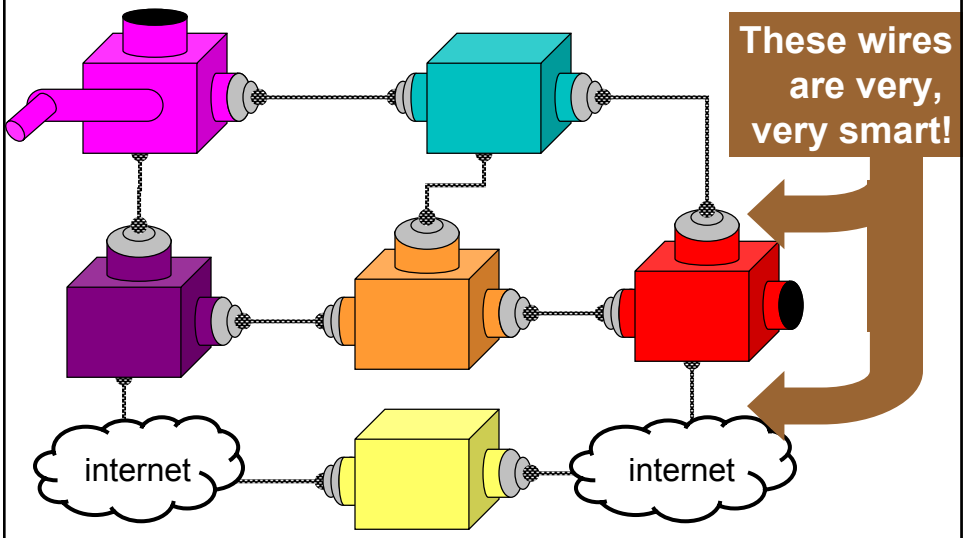
Imagine a company migrates to a new system, OS, etc.



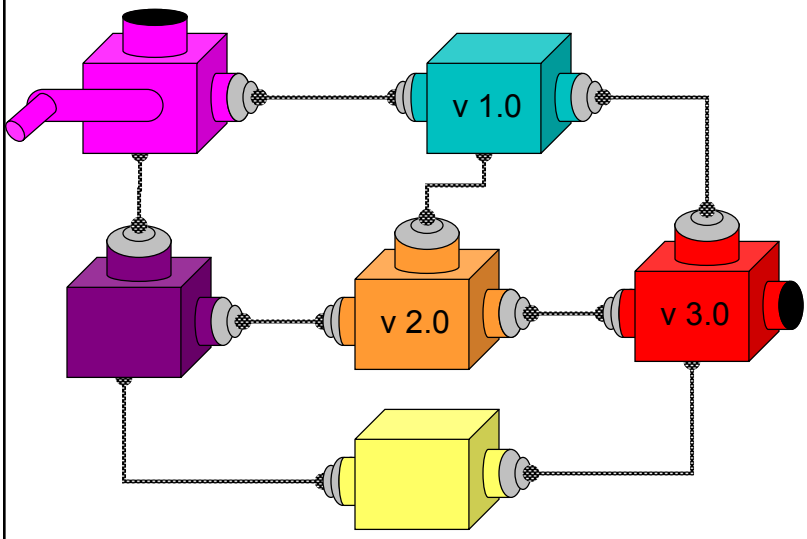
What if the source to this one part is lost???

12

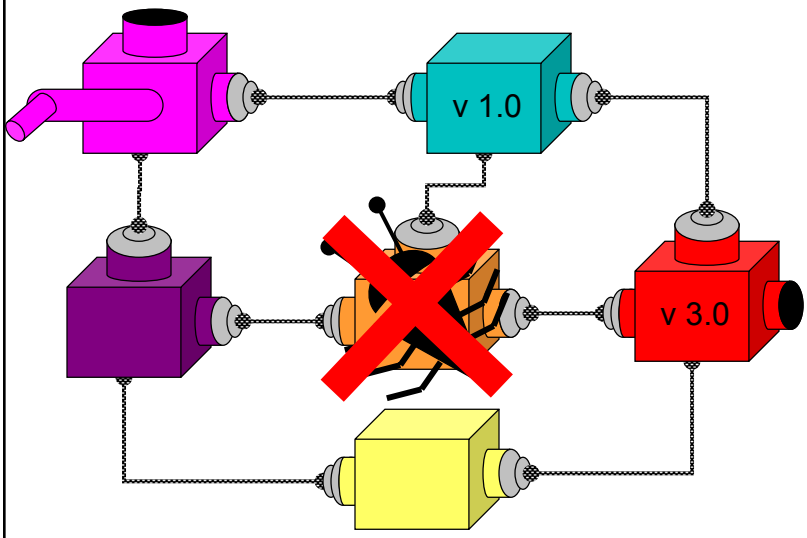
Transparent Distributed Computing



3. Incremental Evolution With Multiple 3rd party software

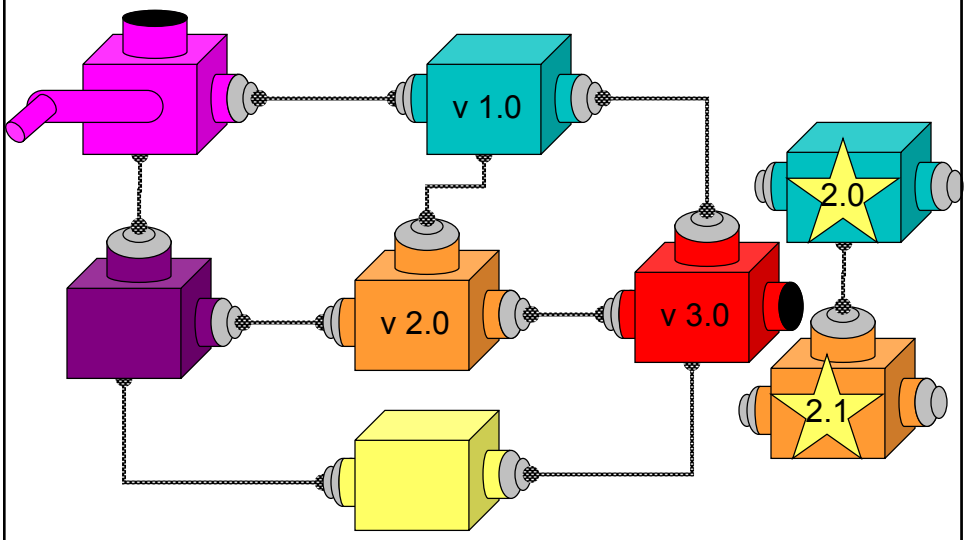


Now suppose you find this bug...



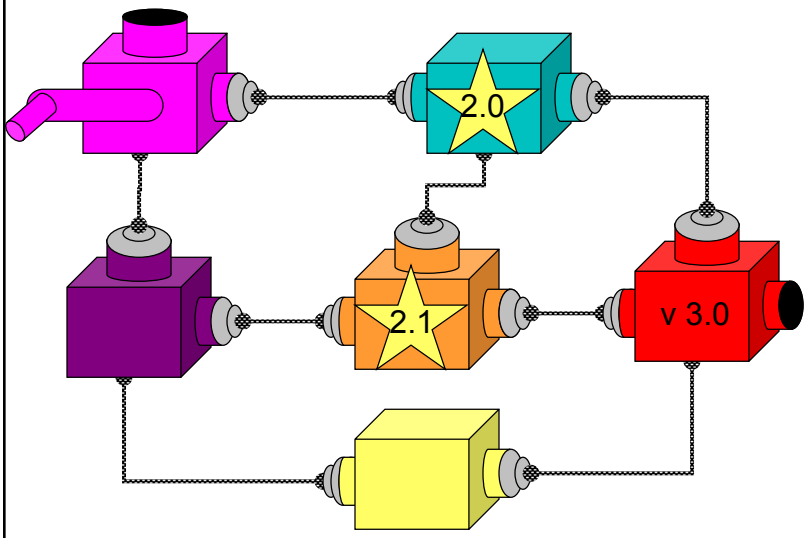
15

Good news: an upgrade available
Bad news: there's a dependency



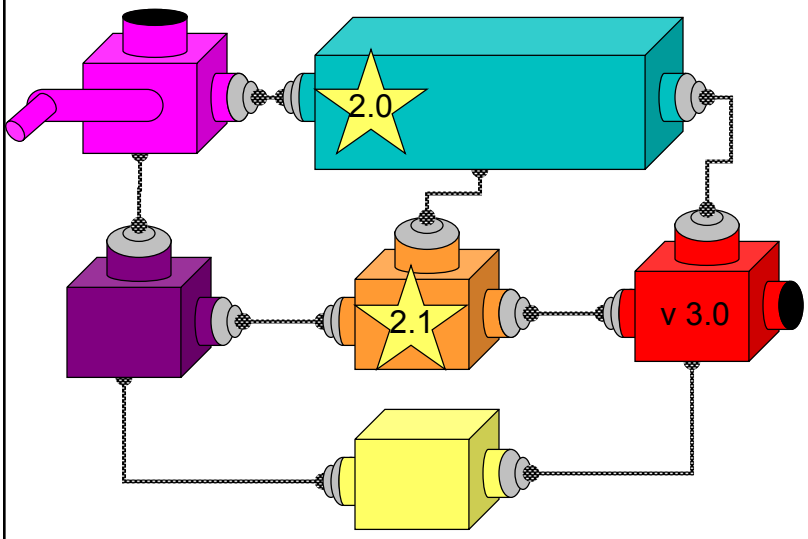
16

Great News:
Solvable with Components



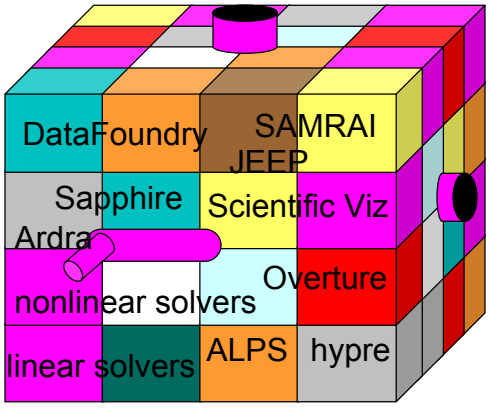
17

Great News:
Solvable with Components



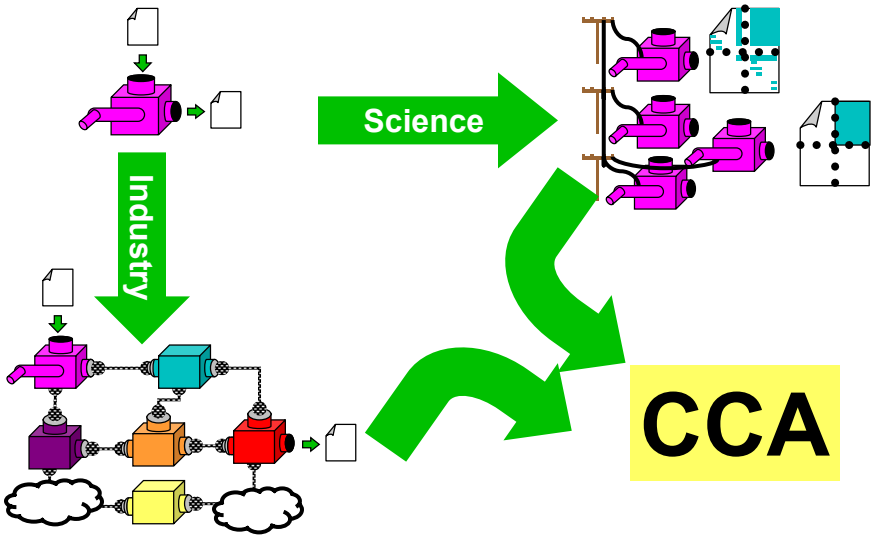
18

Why Components for Scientific Computing → Complexity



- Interoperability across multiple languages
- Interoperability across multiple platforms
- Incremental evolution of large legacy systems (esp. w/ multiple 3rd party software)

The Model for Scientific Component Programming



The End

Next: Intro to Components