1. Onward!: Panel

New Programming Constructs Beyond Inheritance, Patterns, and Notation:

What's left?

William Cook, Phd

CTO, Allegis Corporation

2. Where are we?

Not on a gentle slope of progress

- Reality is more like chaotic experimentation
- Objects are important, but not everything
- No clear consensus on where to go next

Some suggestions...

3. What's Left?

Postmodernism

- Collage of paradigms

Descriptive Language

- Quality of description

Linearity

- Change in requirements ∝ Change in implementation

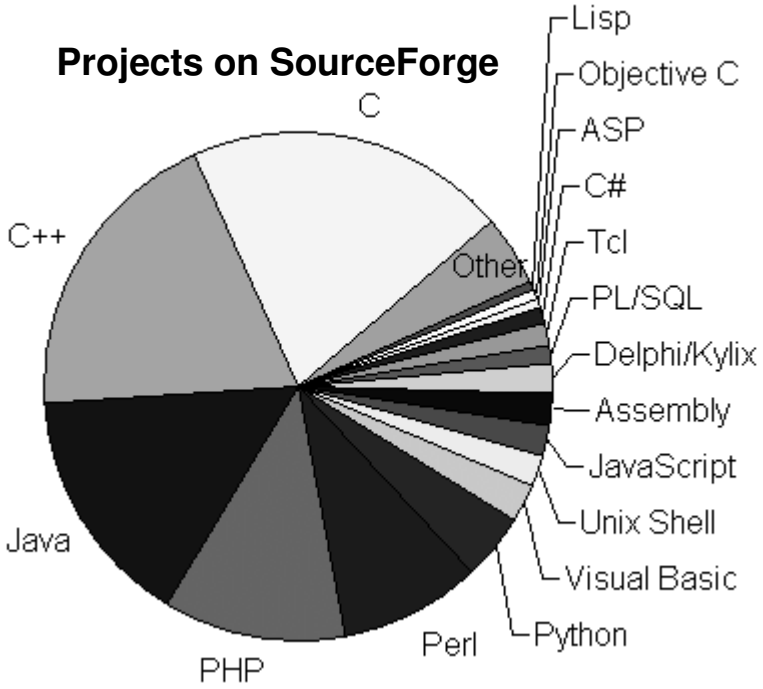
Domain-Specific Modeling

- Are components / libraries enough?

Culture

- Do the big work, *and the small*

4. Postmodernism



Keys

- **Human, not idealized**
- **Reject overall *narrative***
  - Everything is an object.
  - Objects model the real world

Collage of paradigms

- **Make the pieces fit together**

Examples

- **Adding regular expressions to a language**
  - Can do with with classes, but not truly integrated
  - Compilation? Binding variables?
- **Relational Model and Object-Oriented Programming**
  - Still don't have them working together well
- **Allegis**
  - Configurable workflow processes, user-defined classification, roles, targeting
  - Declarative user interface, security policies, declarative data model, event/action model
  - HTML, JScript, C++, declarative transactions, Java C#, IDL, SQL, make, Excel, Outlook

5. Descriptive Language

Some say

- **Objects model the real world**

No...

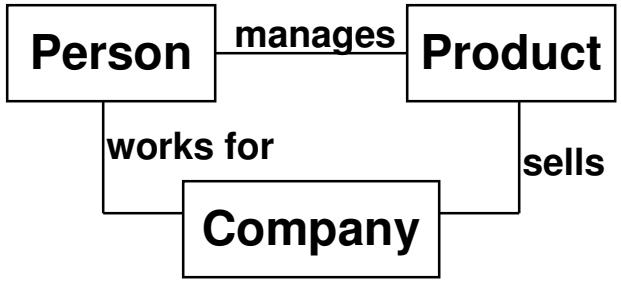
- **Encapsulated state+behavior is one way to model concepts**
  - Concepts are in your head, may or may not be aligned to real world
  - The “way” may or may not be appropriate (Sapir-Whorf)

Instead ask...

- **Does program *describe* things that matter in a way that makes sense?**

Examples

- **Cross-object constraints**
  - Where do I implement “The person who manages a product must work for the company that sells the product”?



- **Swing**
  - Is a Java Swing program the best way to *describe* a user interface?

6. Linearity

Linearity

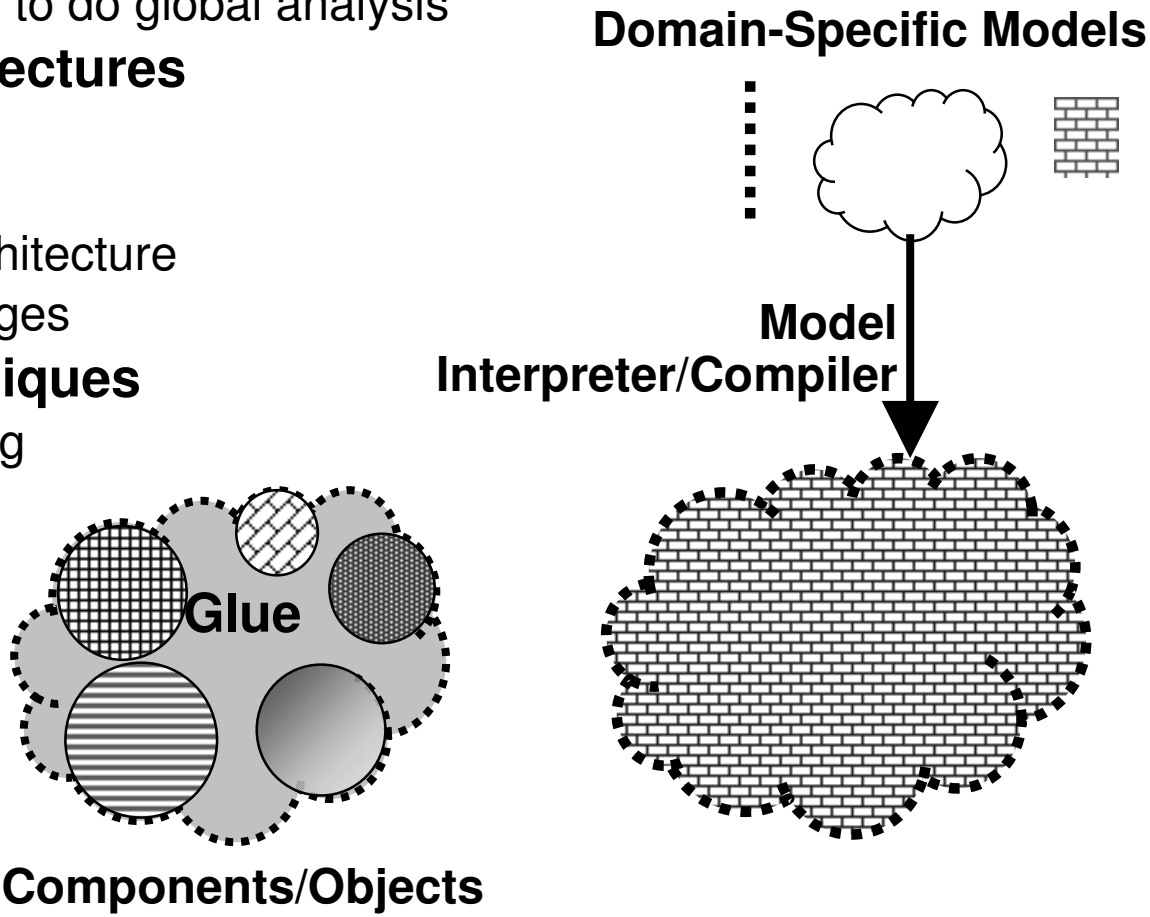
- **A change in requirements is proportional to the change in implementation [Sussman]**
  - Or… program can be refactored simlar change is proportional next time
- **More important than encapsulation, modularity, reuse**

Examples

- **Aspects**
  - Localizing global policies
  - Aspects identify a good problem
    - But is pattern-matching and wrapping code the right solution?
- **SQL**
  - Small change in query results in large change in query plan
    - who cares, because it is automatic

7. Domain-Specific Modeling

- **Benefits**
  - Models can provide descriptive language, locality
  - Reuse the machines that make the parts, not the parts
  - More abstraction, ability to do global analysis
- **Languages and Architectures**
  - Markup languages
  - Precise UML
  - OMG Model-Driven Architecture
  - Domain-specific languages
- **Implementation Techniques**
  - Generative programming
  - Meta-programming
  - Staged computation
  - Macros
- **The next big thing**



8. Culture

Do the big work, *and the small*

- **Make the basic things trivial**
  - Web and XML are simple ideas with great impact
  - Look for incremental improvement in addition to revolutionary ideas
- **Then solve the hard problems**

Academia & Industry working together

- **Industry needs help now, not just in 10 years**
- **Need more mutual understanding**
- **No more Colored Points**
- **Consider Academic and industrial value systems**

9. Summary

Postmodernism

- Making paradigms work together is hard

Descriptive Language

- Does program *describe* things that matter in a way that makes sense?

Locality and Linearity

- Architecture should localize things that are important

Domain-Specific Modeling

- “Everything is a model”

Culture

- Do the big work, *and the small*