

# ***Previous Proposal and Critique: The CC 2001 PL Unit***

PLC Workshop, Boston  
May, 2008

Gary T. Leavens  
University of Central Florida  
[www.eecs.ucf.edu/~leavens](http://www.eecs.ucf.edu/~leavens)

# Outline

---

- ❑ (History)
- ❑ Proposal Details
- ❑ Discussion

# *History*

---



# *History*

---

- Fall 1998
  - Outline of topics didn't include PL.
  - Some PF units seem to be advanced PL.
  - General uproar
- July 22, 1999, KFG formed.
- August 6, 1999, KFG preliminary report.

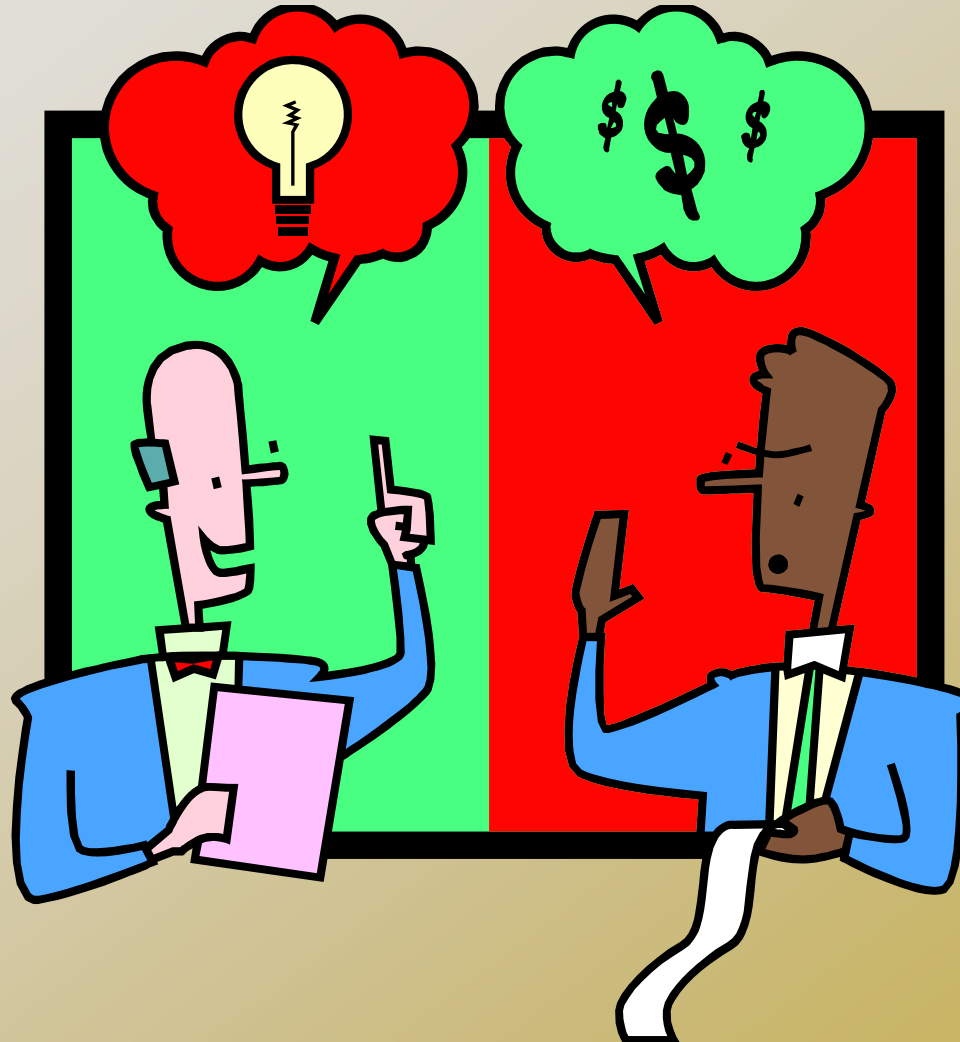
# *Original PL KFG*

---

- ❑ Kim Bruce (was at Williams)
  - ❑ Benjamin Goldberg (New York U.)
  - ❑ Gary T. Leavens (was at Iowa State U.)
  - ❑ John Mitchell (Stanford)
  - ❑ Chris Haynes (Indiana)
  - ❑ Joseph Hummell (U. of Illinois, Chicago)
- Also had comments from others.

# *The Proposal*

---



# *Summary of Changes from CC '91*

---

- ❑ Logic programming out of core.
- ❑ Much less on automata, grammars.
- ❑ More on modules and information hiding.
- ❑ More detail, esp. for OO units.
- ❑ Rearranged to increase coherence.
- ❑ 10.5 fewer hours, total core hours: 35.5

# Knowledge Units

## *Proposed / Accepted*

---

- PL1: History and Overview of PLs (1.5/.5)  
PL1: Overview of PLs (2)
- PL2: Virtual Machines (1)  
PL2: Virtual machines (1)
- PL3: Formal Languages, L. Analysis (1.5/.5)  
(0)
- PL4: Language Translation Systems (1.5/1/1)  
PL3: Intro. to language translation systems (2)  
PL8: Language translation systems (0/+)

# Knowledge Units

## *Proposed / Accepted*

---

- PL5: Types (5/0/1)  
PL4: Declarations and types (3)  
PL9: Type systems (0/+)
- PL6: Control of execution (3/1)  
(0)
- PL7: Declarations and Modules (5)  
PL5: Abstraction mechanisms (3)
- PL8: Run-time Storage Management (3)  
(0)

# *Knowledge Units*

## *Proposed / Accepted*

---

- PL9: PL Semantics (2/0/3)  
PL10: PL semantics (0/+)
- PL10: Functional Programming (5)  
PL7: Functional programming (0/+)
- PL11: OO Paradigm (4/0/1)  
PL6: OO programming (10)
- PL12: Distributed and Parallel (3/2/2)  
(0)

# *Knowledge Units*

## *Proposed / Accepted*

---

- (0)  
PL11: PL design (0/+)

# *PL1: History and Overview*

## *PL1: Overview of PLs*

---

Core lecture topics: (1.5 hrs / 2 hrs)

- ❑ History of programming languages
- ❑ Brief survey of programming paradigms
  - ❑ Procedural languages
  - ❑ Object-oriented languages
  - ❑ Functional languages
  - ❑ Declarative, non-algorithmic languages
  - ❑ Scripting languages
- ❑ The effects of scale on programming methodology

*Optional lecture topics :*

- ❑ Parallel Programming paradigms

# *PL2: Virtual Machines*

---

## *Core Lecture Topics: (1 hr)*

- ❑ The concept of a virtual machine
- ❑ Hierarchy of virtual machines
- ❑ Intermediate languages
- ❑ Security issues

# *PL3: Formal Languages and Language Analysis*

---

## *Core Lecture Topics: (1.5 hr)*

- ❑ Overview of regular expressions, context-free grammars, and use.
- ❑ Parse trees, ambiguous grammars.

# *PL5/PL4: Declarations and Types*

---

*Core Lecture Topics: (5 hrs / 3 hrs)*

- ❑ Types as set of values + operations
- ❑ Declaration models  
(binding, visibility, scope, and lifetime)
- ❑ Overview of type checking
- ❑ User-defined types
- ❑ Parametric polymorphism (Generics)
- ❑ Subtype polymorphism
- ❑ Garbage collection

# PL5/PL9: Type Systems

---

## *Optional Topics:*

- Data type as set of values + operations
- [Type constructors] ( $\rightarrow$ ,  $+$ ,  $\times$ , ...)
- Type checking models
- **Semantic models of** user-defined types
- Parametric polymorphism
- Subtype polymorphism
- Type-checking algorithms

# *PL6: Control of Execution*

---

## *Core Lecture Topics: (3 hrs)*

- ❑ Expressions, order of evaluation of sub-expressions
- ❑ Statements (conditional, procedures, iterators)
- ❑ Exceptions and exception handling

## *Intermediate Lecture Topics: (1 hr)*

- ❑ Parallel composition ( $S1 || S2$ )
- ❑ Functions delay evaluation

# ***PL7: Declarations and Modules***

## ***PL5: Abstraction Mechanisms***

---

*Core Lecture Topics (5 hrs / 3 hrs)*

- ❑ **Declarations** (binding, visibility, lifetime)
- ❑ **Procedures, functions, and iterators**
- ❑ **Parameterization mechanisms**
- ❑ **Mechanisms for sharing and restricting visibility**
- ❑ **Activation records and storage management**
- ❑ **Type parameters and parameterized types**
- ❑ **Modules in programming languages**

# *PL10 / PL7: Functional Programming*

---

*Core / Optional Lecture Topics: (5 hrs / 0 hrs)*

- ❑ Overview and motivation
- ❑ Recursion over lists, natural numbers, trees, and other recursively-defined data
- ❑ Pragmatics: debugging by divide and conquer; persistency of data structures
- ❑ Amortized efficiency for functional data structures
- ❑ Closures, and uses of functions as data

# *PL11 / PL6: OO Paradigm / Programming*

---

*Core Lecture Topics: (4 hrs / 10 hrs)*

- ❑ Overview and motivation
- ❑ Object-oriented design
- ❑ Encapsulation and information-hiding
- ❑ Separation of behavior and implementation
- ❑ Mechanisms for defining classes and interfaces
- ❑ Classes and subclasses
- ❑ Object creation and initialization
- ❑ Inheritance (overriding, dynamic dispatch)
- ❑ Polymorphism (subtype polymorphism vs. inheritance)
- ❑ Class hierarchies
- ❑ Collection classes and iteration protocols
- ❑ Internal representations of objects and method tables

# *PL12: Distributed and Parallel Programming Constructs*

---

## *Core Lecture Topics: (3 hrs)*

- ❑ Overview and motivation
- ❑ Explicit communication primitives
- ❑ Communication primitives for models with shared memory

## *Optional Topics:*

- ❑ Programming primitives for data-parallel models.
- ❑ Comparison of language features for parallel and distributed programming.
- ❑ Optimistic concurrency control vs. locking and transactions
- ❑ Coordination languages (e.g., Linda)
- ❑ Asynchronous remote procedure calls (pipes)
- ❑ Other approaches (functional, nondeterministic)

# *What Didn't Work*

---

- ❑ Semantics of standard features as core
- ❑ Including as core topics what programmers and engineers don't use (e.g., functional programming)
- ❑ Core topics justified by progress in PLs or further study of PLs

# *What Did Work*

---

- Justifications from viewpoint of programmer or (software) engineer
- Suggestion of advanced/optional topics

# *Discussion*

---



# *Discussion Questions*

---

- ❑ What PL skills & experiences needed by programmers and software engineers?
- ❑ How much semantics do they need?
- ❑ What's the role of: IDEs? Libraries?
- ❑ Can we make common cause with PF, SE, and OS?

# *Discussion Questions*

---

- ❑ What omitted facets should we push for?
- ❑ What aspects of CC 2001 were better?
- ❑ What did everyone miss?
  
- ❑ What kinds of topics did CC 2001 take?
- ❑ What kinds did they omit?
- ❑ Can we draw any lessons from that?

# *Discussion Questions*

---

- How can we distinguish what's a current hot topic vs. of enduring value?
  - Aspect-oriented?
  - Model-driven development?
  - Service-oriented architecture?
- Should we blend in current hot topics?