+’s to Wegner’s Milestones

- Languages
  - **Smalltalk -** first truly object-oriented language
  - **Gypsy -** demos automated verification is feasible in parallel language
  - **CLU -** first to demo utility of data abstraction
  - **FP -** functional languages come into being.
  - **CSP -** clarified many communication/ synchronization issues in parallel langs
  - **Ada -** whether of not you like it, it's a significant accomplishment
  - **Logo -** computing for children is possible
  - **Mesa -** static checking isn't mandatory in parallel languages
  - **SETL -** first very high level language
  - **Prolog -** demonstrates feasibility of logic programming

+’s to Wegner’s Milestones

- More Languages
  - **C++** - virtual functions & static type checking
  - **Java** - It is possible to improve an existing language???
  - **Visual Languages**???
  - **Unity**?
- – Perl
- – ML
- – Why not C???

+’s to Wegner’s Milestones

- Concepts
  - denotational semantics
  - specification languages
  - verification successes
  - object-oriented programming,
  - (class instances and inheritance issues)
  - logic programming
  - functional programming
  - data flow
  - parallelism (synch and comm)
  - nondeterminacy
  - models of computation

+’s to Wegner’s Milestones

- More Concepts
  - Exception handling
  - Real-time languages
  - Polymorphism
  - Ada parameter passing
  - Managing reference variables
  - Expanding notion of what a PL is
  - Visual languages
  - Simulation
  - Operating systems
  - Databases
  - Language environments
  - Type inferencing

- closures
- subtyping

+’s to Wegner’s Milestones

- Implementation Ideas
  - syntax directed compiling
  - parser/lexical generators
  - programming environments
  - syntax directed editors
- Architectural Influences
  - micro-computers (VLSI and its impact on language design)
  - interactive programming capabilities
  - language-based architectures (Lillith, LISP machines, Transputer)
  - parallel architectures
  - Virtual memory
  - RISC;   RISC/CISC

Dijkstra: Threats to Computing Science

“Not getting lost in the complexities of our own making and preferably reaching that goal by learning how to avoid the introduction of those complexities in the first place. <u>That</u> is the key challenge computing science has to meet.”

- p.3

“…the suggestion that the proposed style of composing iteratively would save time is an obvious and blatant lie.”

- p.7

Dijkstra: Threats to Computing Science

"It is not only the performing artist who is, in a very real sense, shaped by the instrument he plays; this holds as well for the Reasoning Man…"

- p. 8

"The quest for the ideal programming language and ideal man-machine interface that would make the software crisis melt like snow in the sun had -- and still has! -- all the characteristics of the search for the Elixir and the Stone."

- p. 10

Dijkstra: Threats to Computing Science

"…All we needed was 'intelligence amplification'…they have probably discovered it would amplify stupidity as well…"

- p.12

"…the public…tends to confuse…the composing of a symphony with the writing of its score."

Dijkstra: Threats to Computing Science

"…the programmable computer is no more and no less than a handy device for the implementation of any thinkable mechanism. As such it poses on us the burden to demonstrate which mechanisms we can think of sufficiently clearly. It implies the challenge of blending engineering with the techniques of scientific thought; this challenge is exciting and we are ready for it."