Programming-in-the-Large Versus Programming-in-the-Small

Charles Shelton
17-939
October 10, 2001

DeRemer & Kron 76

- Make the claim that large software systems are fundamentally different from small software systems
  - Structured programming isn’t enough to design large software
  - Benefits of modular decomposition become subsumed by need for complicated glue code to connect modules

- Propose developing a “module interconnection language”
  - Specifically deal with large system structure
  - Task of connecting modules into a large system is a separate problem that requires its own tool

- A useful Module Interconnection Language should aid:
  - Project Management
  - Design
  - Communication
  - Documentation
Impact

◆ This paper didn’t have much in the way of validation
  • Argument about why MIL’s are necessary and what they could accomplish
  • Discussion about MIL 75 language semantics don’t really apply anymore

◆ Ideas about large software systems carried into software architecture
  • Parnas’ Modular Decomposition paper
  • Need separate abstraction model for overall system structure
  • Analyze the system in terms of its structure above the module level
  • Architecture Description Languages

Research Question

◆ Practical Problem:
  • Can we build large software systems better by exploiting the properties of the system’s overall structure to achieve better reliability?

◆ Research Problem:
  • Can we build a separate tool for connecting software modules into a large system that is easier to design and maintain?

◆ Research Plan:
  • Can large system design be done better?
  • Devise technique of module interconnection languages
  • “Look, it SHOULD work!”
  • Argument, but no evidence presented here
Pro Forma Abstract

• Radical Solution (RS)
  • A radical solution to the problem of building large software systems is described, based on using a module interconnection language to specify system structure. In comparison with structured programming and modular decomposition alone, it offers the possibility of a system that is easier to design and maintain because module development is separated from the task of connecting the modules together. The side effect is that the module interconnection language must have a complex compiling system to be useful in designing the software system.