Problem: Teaching in the Small

• How important is the role of examples & associated models in such a course? How important is it to use models of real problems?
  – Even in coding, examples that students see in an undergrad curricula are often tied to small examples that do not highlight the challenges of real development that has different scale
  - Need for deeper scale examples that show challenges of engineering changes in the presence of complex underlying heterogeneous platforms; students may gain appreciation of challenges they were unaware
  - Example contexts
    - Cloud variations, mobile computing platforms, different robot vendors
  - Challenge is implementing this scale in presence of single assignment; perhaps benefits more in case of individual research experience rather than course-based
Highlighting Modeling Benefits
Problem: Teaching Syntax

• **How to overcome the bias some students have against using modeling in software?**
  
  – Many current courses focus on teaching syntactic issues of UML
  – Missing: Topics of creativity, social impact, general abstraction, broader issues of modeling benefit
  – Comparison to new effort in USA for K-12 CS Education
    • Current focus on teaching syntax of Java
    • Introduction of new “CS Principles” course with broader application and understanding of computing
    • Concentrates on “Big Ideas” of Computer Science
    • What are the “Big Ideas” of modeling that motivate its usage in a way that is more appealing than learning about the specific semantics of state diagrams
Modeling for Non-Majors

• Most CS departments teach general CS literacy courses to non-majors
  – Does a similar modeling course also make sense?
• What would a modeling course for non-majors look like?
  – Perhaps not UML; screams for DSMLs
• Similarities and differences:
  – What are the things taught in this class that would not be taught in a similar course for CS majors?
  – What things would be taught in this class that would also be taught in a similar course for CS majors?