**Why we use math in Software Engineering? P.3Last**

Which Mathematics for Software Engineers? Functions, Relations, and Sets , Basic Logic, Proof Techniques (direct, contradiction, inductive), Basic Counting, Graphs and Trees, Discrete Probability, Finite State Machines, regular expressions, Algorithm Analysis, Number Theory, and Algebraic Structures. Foundational core material included: Abstraction: Generalization, Levels of abstraction and viewpoints, Data types, class abstractions, generics/templates and Composition. Modeling: Principles of modeling, Pre and post conditions, invariants Mathematical models and specification languages, Model development tools and model checking/validation, Modeling/design languages (such as UML, OOD, and functional), Syntax vs. semantics (including understanding model representations), and Explicitness (make no assumptions or state all assumptions) Though not explicitly within the core, a year of calculus was also needed to: enhance mathematical maturity and thinking, provide a contrast with discrete mathematics concepts, and guarantee enough background for various client and application disciplines. “Statistics and empirical methods were also recommended, personally, i would also add linear algebra. Advanced mathematical courses, including graph theory, combinatorics, theory of computing, probability theory, operations research, and abstract algebra, may additionally be required, betting on the goals of the program and therefore the needs of individual students. Mathematically oriented software engineering courses these days cover formal specifications, formal methods, mathematically rigorous software design, software verification and validation, and software models and model checking.” **Conclusion** Mathematical reasoning is intrinsic to each traditional engineering and software engineering, though every discipline uses totally different foundational mathematics. traditional engineers use continuous mathematics primarily in a very calculational mode for modeling, design, and analysis, including to calculate, say, load on a bridge component, compute the wattage of a resistor, or verify the optimum weight of an automobile suspension system. software engineers sometimes use discrete mathematics and logic in a declarative mode for specifying and verifying system behaviors and for analyzing system features. The RC circuit and iteration invariants represented earlier illustrate basic mathematical reasoning. However, engineering is significantly deeper and broader than these examples indicate. Engineers are systems architects who perceive and apply the foundational principles of the discipline. software engineers should therefore learn to use mathematics to: construct, analyze, and check models of software systems, compose systems from components, develop correct, efficient system components, specify (precisely) the behavior of systems and components, and analyze, test, and evaluate systems and components. One area wherever traditional engineering has an advantage is that the number, variety, and maturity of tools for mathematical modeling, design, analysis, and implementation, together with standard languages for communication in blueprints and schematic circuit diagrams and computer-aided prototyping, design, and analysis tools. Surveys of current practices reflect reality, several software engineers haven't been tutored to use discrete mathematics and logic as effective tools. Education is the key to making sure future software engineers are ready to use mathematics and logic as tools for reasoning and thinking.

STD : Abdulrahman Alghamdi Words : 459

ID : 442051025 Quote rate : %16.5