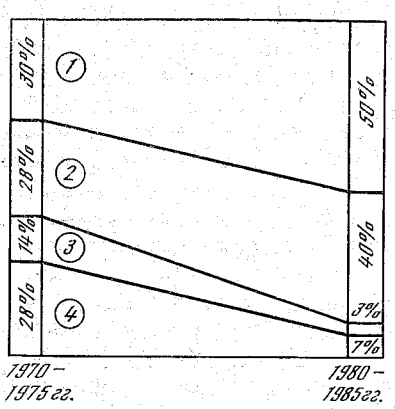

Рис. 24. Динамика структуры трудозатрат по фазам проекта в 1970—1980 гг.

1 — сопровождение и поддержка,
2 — анализ приложений и постановка задачи,
3 — кодирование и автономная отладка,
4 — комплексная отладка и приемосдаточные испытания

Рассчитано по: 01 Informatique-Hebdo, 1980, 14, IV, p. 35; данным фирмы «TRW» (приводится в: Зарубеж. радиоэлектрон., 1974, № 12, с. 9)

рования — автоматический синтезатор любых программ (т. е. повысить производительность труда программистов, измеряемую в строках кода в день, от существующего уровня в 100% за 10 лет до «бесконечности» или предела, определяемого темпом загрузки в этот «автомат» спецификаций на программы), то это почти не снизило бы трудоемкость выполняемых проектов (уменьшение трудоемкости составило бы 15—20% для традиционных областей приложений и 3—5% для наиболее массовых в настоящее время областей приложений ЭВМ).

Иными словами, в настоящее время, когда основные усилия профессиональных программистов затрачиваются на то, чтобы понять, что должна делать программа, а не на то, как ее закодировать, измерять производительность труда программистов строками кода в день — почти то же самое, что измерять производительность труда инженера-конструктора числом линий, проведенных им за день на чертеже.

Если нелепость этого примера очевидна (понятно, что здесь важна эффективность создания конечной продукции), то чем же тогда объяснить, что до сих пор производительность труда программистов оценивается толщиной колоды?

Субъективная причина понятна. Есть определенная логика в таком способе измерения. Если эффективность программирования как производственной деятельности определяется числом строк программного кода, которые может выдать в день профессиональный программист, значит, проблема кодирования, на которой построено все здание большой науки программирования, и есть самая важная из профессиональных задач.