

## Software bugs cost the U.S. economy \$59.5 billion annually

(A study by US Commerce Department's National Institute of Standards and Technology)

The same study also says:

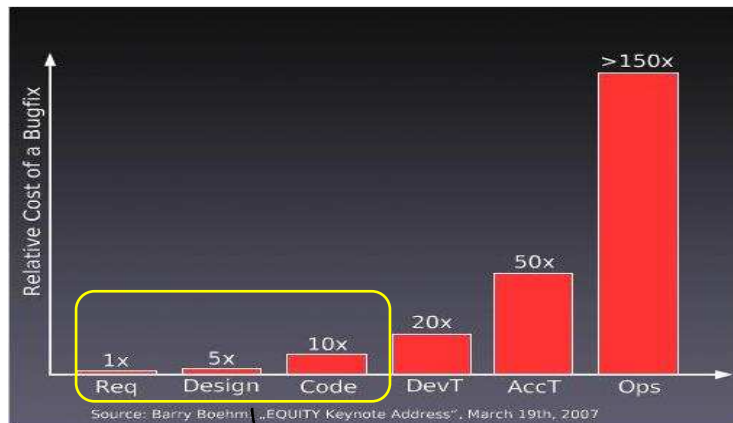
**More than a third of these costs, or an estimated \$22.2 billion, could be eliminated by an improved testing infrastructure that catches errors closer to the software products' developmental stages where errors are introduced.**

However, the study goes on to point out that:

**More than half of software bugs are not discovered until well into the development process or until post-sale use.**

It is well-known that as we go down the software-life cycle, the cost of fixing a bug increases drastically.

Still, all attempts to improve Software Quality focus on **"FINDING AND FIXING DEFECTS"**



*What if 90% of the bugs are caught and fixed in this stage?*

IN ADDITION TO A ROBUST WAY TO FINDING AND FIXING DEFECTS, WHAT IS NEEDED:

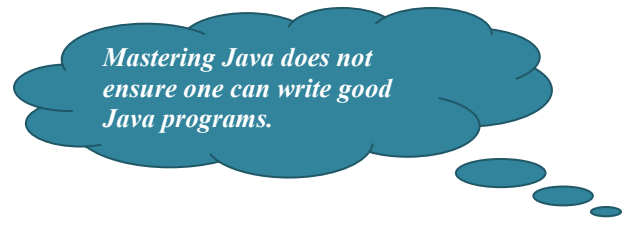
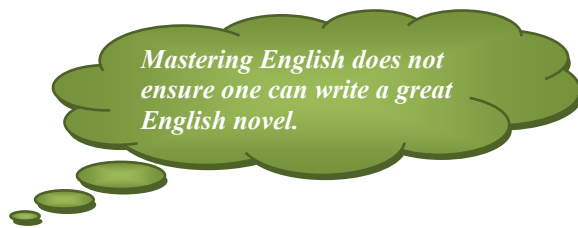
- A WAY TO PREVENT BUGS FROM BEING BORN ...
- A WAY TO INCREASE PERSONAL EXCELLENCE OF EACH INDIVIDUAL PROGRAMMER ...

*“The top software developers are more productive than average software developers not by a factor of 10x or 100x or even 1000x, but by 10,000x.”*

*- Nathan Myhrvold, former CTO, Microsoft*

Increasing a software developers’ productivity is often equated to increasing technical skills OR implementing/adhering to better processes.

However, that is just one part of the equation ...



A programmer needs many more skills than just mastering technical skills and processes.

<b>Knowledge</b>	Recall the learning on technical skills and processes.	<ul style="list-style-type: none"> <li>• Syntax of programming languages/libraries</li> <li>• Problem-domain knowledge</li> <li>• Knowledge of software development processes</li> </ul>
<b>Comprehension</b>	Understand the meaning and interpretation of instructions and problems, represent a problem effectively.	<ul style="list-style-type: none"> <li>• Understand the problem to solve</li> <li>• Translate a logic to program</li> <li>• Minimize the number of steps</li> <li>• Comprehend an existing code-base efficiently.</li> </ul>
<b>Application</b>	Use a concept in a new situation or use an abstraction to novel situations in workplace	<ul style="list-style-type: none"> <li>• Apply theoretical knowledge</li> <li>• Smart problem-solving skills</li> <li>• Good designing skills</li> <li>• Algorithmic thinking</li> </ul>
<b>Analysis</b>	Separate materials or concepts into component parts so that its organizational structure is understood.	<ul style="list-style-type: none"> <li>• Understand and articulate complex requirements</li> <li>• Understand the architecture of large applications</li> <li>• Understand the logic of complex programs</li> <li>• Understand the clues while debugging a program</li> </ul>
<b>Synthesis</b>	Build a structure or pattern from diverse elements, put together to form a whole, create new structure/meaning	<ul style="list-style-type: none"> <li>• Find the correct logic of a program</li> <li>• Evaluate the program logic</li> <li>• Ability to implement small modules independently</li> <li>• Integrate a large number of complex modules</li> </ul>
<b>Evaluation</b>	Make judgments about the value of ideas or materials	<ul style="list-style-type: none"> <li>• Ability to describe a program-logic</li> <li>• Ability to evaluate and justify a program logic</li> <li>• Debug a program logically and systematically</li> <li>• Take smart design decisions</li> </ul>

All the skills described above can be put together as **PROGRAMMING QUOTIENT**.

**Artistic Programming** is aimed at helping programmers enhance their **PROGRAMMING QUOTIENT** by training individuals in developing a systematic thought-process towards understanding, analyzing and solving of complex problems.

**Artistic Programming** impacts all aspects of the Software Development Lifecycle:

1. **Requirement:** Artistic Programming trains an individual in smart problem-solving skills. Good problem-solving begins with the right understanding, articulation and representation of the problem-statement. This skill helps the programmers to collate, interpret and represent requirements effectively.
2. **Design:** Artistic Programming helps the programmers create designs which are simple, elegant, easy to verify, easy to communicate, easy to implement. This automatically leads to high quality and high productivity.

*What Donald Knuth had to say about Design:*

*"Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do."*

3. **Coding:** Artistic Programming trains the individuals in systematically translating logic into code in an agile, modular manner while ensuring that every line of code is unit-tested. It also helps the programmers manage complexity; manage change in requirements/code, etc.
4. **Debugging:** *Did you know that?*

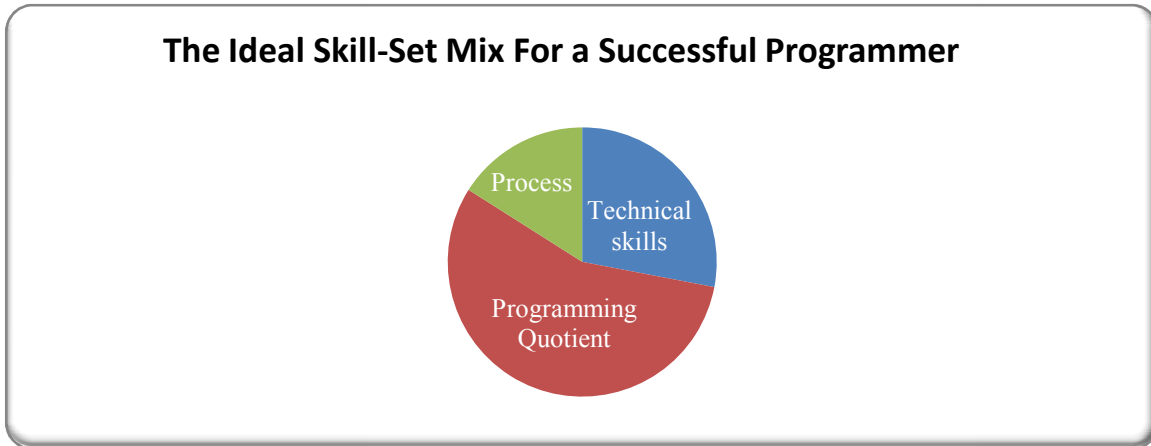
*An average programmer spends only 5% of time in writing new code and 95% in debugging the already written code.*

Artistic Programming trains individuals in mastering systematic approach to debugging. These approaches to debugging have reduced debugging time from several days (including nights) to a few hours.

5. **Team-Work:** Artistic Programming helps programmers view their work in the context of the complete perspective of the software project. It helps programmers understand the rationale behind clients' decisions, managers' strategies, and helps the teams to adopt processes voluntarily. It helps team members communicate easily, effectively and powerfully, impacting team performance.
6. **Productivity :** A streamlined thought-process, good design, coding and debugging skills, a positive approach to the job, all come together to an increased productivity of individuals as well as teams.

Processes and Technical Skills are not Enough.

You need to rev up your **Programming Quotient** through **Artistic Programming**



“With the software in production, fixing bugs is akin to repairing a car while it is driving down the road, long after it has left the drawing board, the assembly line, and the dealer lot. It’s as expensive to do as it can possibly be”.

Forget about finding/fixing defects. Prevent the bugs from being born !!!

