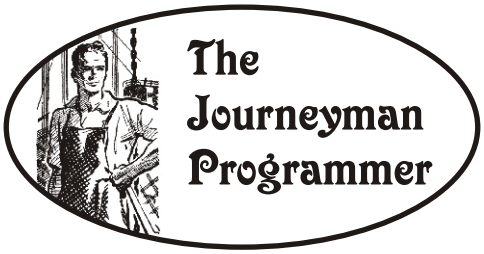

Programming Without Fear

Best Practices for New and Existing Software

When: **November 13 & 14, 2010**
Where: MIT Building E51,Cambridge, MA
Cost: $495 through October 13
$545 October 14 through October 31
$595 after Nov. 1 through Nov. 10
$695 after Nov. 10

Speaker

Gil Broza helps organizations, teams and individuals implement high-performance Agile principles and practices. His coaching and training clients – over 1,000 professionals in 20 companies – have delighted their customers, shipped working software on time, increased their productivity and decimated their software defects.
Beyond teaching, Gil helps people overcome limiting habits, fears of change, blind spots and outdated beliefs, and reach higher levels of performance, confidence and accomplishment.

Who Should Attend

This seminar is designed for *software practitioners, programmers, and designers*. It is best suited to moderately experienced programmers, typically with 2-10 years of software development experience.

This seminar is a production of the Greater Boston Chapter of the ACM

Writing new code is one of the joys of programming. Modifying existing code - legacy code - is often one of the nightmares of programming. *Unfortunately, you will spend more time modifying existing code than writing new code.*

Modifying existing code often results in **Whac-A-Mole Programming** - you make a change in one part of the code, and a problem crops up in another place. You fix that problem and two more pop up. You quickly find yourself trapped in debug hell!

There is a better way - a sane approach to dealing with both new code and legacy code. A framework for bringing order to code chaos, making changes to existing code that are actually *improvements.* This seminar provides a powerful approach to developing and modifying code.

If you've been programming professionally even just a couple of years, you've likely noticed that:

- **Fixing bugs** can take anywhere from a few minutes to a few days - or even longer! - and is even less fun in code written by other people.
- The **larger** a feature gets, the **longer** it takes to enhance or maintain (and the scarier it gets to do so).
- However **elegant, clear or clever** your code appeared when you first wrote it, when you revisit it a year later you shudder (or scratch your head).
- Code-level documentation, even when it exists, somehow never gives you enough information.
- Even **"tested"** code breaks where you least expect it to.

Only a decade ago this was accepted wisdom about our profession. But *it doesn't have to be this way!*

In this **Legacy Code Without Fear** seminar, you will learn reliable, sustainable and enjoyable software development practices which you can immediately put to use. These techniques, first popularized in Extreme Programming (XP), have been proven, refined and extended for more than a decade. By using them you will produce working code faster, increase its quality, and reduce technical debt — the demon that snarls later development. After this seminar you will be able to:

- Develop simple, clear, tested code faster than you ever did
- Practice modern techniques such as test-driven development and refactoring
- Evolve object-oriented software with tests' guidance
- Unit-test the tough cases

… And have the skills needed to stop dreading legacy code!

Details and registration: www.gbcacm.org

Journeyman Program

GBC/ACM is implementing a program to meet the needs of Journeyman Programmers. This seminar is the first in a series that deal with real world issues encountered by mid-career software development professionals.

Journeyman Programmers

Journeyman comes from the world of guilds and crafts, where workers were classified as *apprentice, journeyman, or master.* An **apprentice** was the lowest rung, someone who was being trained. An apprentice required close supervision on most tasks, and could do little by themselves.

A **master** was an expert in his craft, qualified to do everything, supervised journeymen, and often worked independently.

The **journeyman** was in the middle. They had mastered the basics of the craft and were able to work with minimal supervision. However, they still required some **direction**, were developing expertise and knowledge, and were not ready to work independently.

Loosely defined, a *Journeyman Programmer* is someone with 2-10 years of industry experience. They have mastered the basics of coding, and can do productive work. However, they have not mastered the big picture, are still learning, and can improve their skills in many areas.