Proactive Testing™ not only can reduce defects in delivered software systems, it also can cut development time and cost. Unlike conventional testing, which merely reacts to whatever has been designed/developed and frequently is perceived as interfering with development, Proactive Testing™ also helps prevent problems and make development faster, easier, and less aggravating.

Moreover, Proactive Testing™ can reduce project budget and schedule overruns by spotting the commonly-overlooked up to 75 percent of showstopper problems which are major sources of late, unplanned significant redesign and rework. Organizations that learn to *let Proactive Testing™ drive development* not only can avoid such showstoppers' impacts, they also can significantly reduce the development time and effort needed to correct the errors that cause the showstoppers.

Proactive Testing™ incorporates and goes beyond proven conventional testing techniques, also enlisting more effective approaches and methods, which make developers, managers, and users, as well as testers become active advocates of using Proactive Testing™. Key Proactive Testing™ concepts and techniques include:

- Find WIIFMs that are meaningful personally to those who are affected, primarily saving them time, effort, and aggravation. This is in contrast to the seldom-well-received "do it because it's good for you" conventional testing orientation.

- Intertwine testing with every development activity, so that each development deliverable throughout the life cycle is tested by appropriate means at the time it's developed. This successively improves the quality of what is developed, so it has fewer problems to start with; and then superior testing catches more of those fewer remaining problems to net a far cleaner end software product.

- Keep User Acceptance Testing and Technical Testing (traditional unit, integration, system, and special tests) independent. This builds users' competence, confidence, and cooperation. It truly double checks from the users' perspective and finds issues missed by conventional techniques, which often mainly have users repeat subsets of system tests already defined and run by the technical organization.

- Emphasize that developers must know how to tell whether what they've developed is correct. This is in contrast to conventional testing's common reliance on users to do much of technical testing, on the premise that "only the users know how the system is supposed to work." If so, then what do the developers know?

- Plan and design tests early, as much as possible in conjunction with System Design. Not only is this more efficient than conventional after-the-fact reactive testing, it also is one of the most powerful ways to spot design issues before mistakes are made coding them.

- Write no more than is helpful, but no less. Use low-overhead formats for test plans and designs to economically prompt and capture discovery of many test conditions that conventional development and reactive testing commonly overlook.

- Prioritize as you go, level-by-level so that scarce time and resources continually are refocused on testing the most important risks. This not only enables the conventional reactive testing strategy of testing the highest risks more, but it also enables testing the higher risks earlier as well.

- Plan and design tests in a manner which promotes reuse. Opportunities for reuse are increased considerably when such planning and design are done early and with respect to both test cases and also test design specifications, which are a valuable technique that most conventional testers are not familiar with.

- Let testing drive development. Feedback information from test planning and design into the development process so that problems not only are prevented, but they are prevented in ways which minimize system redesign and rework. Conventional reactive testing cannot positively influence development in this exceedingly powerful manner.

To help achieve the significant Proactive Testing™ advantages, Go Pro Management, Inc. provides consulting and training assistance, including:

- Our three-day ***Proactive Testing™ of Software and Systems*** (also called ***Winning Approaches to Structured Software Testing***) explains Proactive Testing™ and how it applies to each of the various levels and types of technical testing (as distinguished from user acceptance testing) throughout the full life cycle. Shorter subsets of this information include:
  - ***Proactive Testing™: Risk-Based Test Planning and Design*** (2 days)
  - ***Project Manager's Success Secret: Let Proactive Testing™ Drive Development*** (2 days)
  - ***Developing Software Testing & Quality Assurance Techniques*** (2 days)
  - ***Proactive Testing™ Management*** (1 day)
  - ***Risk-Based Testing*** (1 day)
  - ***Managing the Test Execution Process*** (1 day)
  - ***Developing Reusable Test Designs*** (1 day)

- The following one-day courses describe specific project and process management techniques which effective Test Managers will use as part of Proactive Testing™:
  - ***Test Project Management*** (1 day)
  - ***Estimating and Controlling Testing*** (1 day)
  - ***Meaningful System Measurement*** (1 day)
  - ***Test Process Measurement and Improvement*** (1 day)

- Our one-day ***Proactive User Acceptance Testing*** seminar shows how to plan and carry out tests to demonstrate that the delivered system in fact meets the business requirements. Proactive planning of these tests not only increases user competence and confidence, it also helps detect missing, unclear, and incorrect requirements.

- Our one-day ***21 Ways to Test that Business Requirements Are Right*** (also presented with the title ***Evaluating Business Requirements***) seminar shows a variety of methods which are much more powerful than conventional techniques to review accuracy and completeness of requirements that already have been defined. This content also is presented as the first day of our two-day ***Testing Early in the Life Cycle*** (also presented with the title ***Testing that Requirements and Designs Are Right***) seminar, where the second day presents more than 15 ways to test that designs are right.

- Robin Goldsmith's recent Artech House book, ***Discovering REAL Business Requirements for Software Project Success***, combines and elaborates upon the above seminars' concepts and techniques for both discovering and testing the adequacy of business requirements.

