Draft:
‘Real QA’
[Real Software and System Quality Assurance]
Manifesto

Purpose of this document/manifesto:
Quality Assurance (QA) in software has in fact degenerated into testing alone, on a large scale. Software/IT management has ignorantly allowed this to happen, and it is time for a wakeup call. Of course many parts of the industry have been well-aware of more cost-effective ways of delivering required quality in practice, but this has in fact been largely ignored; while granting very large resources to testing alone (as opposed to smarter upstream engineering practices, based on design, prevention and upstream inspections).

QA Objectives

1. **NO SURPRISES ASSURANCE**: To allow management to understand release consequences fully, in relation to expectations, with the lowest costs, the lowest risks, and the lowest degree of surprises.
2. **MEET EXPECTATIONS**: To make sure that the project investors and sponsors get, at least, what they expect.
3. **BUSINESS PERFORMANCE**: To deliver the ‘business’ (or organisational) results envisaged and promised to project and programme supporters
4. **TECHNICAL PERFORMANCE**: To measure the technical performance (including all Quality attributes) attributes of the system
5. **CONSTRAINT COMPLIANCE**: To ascertain that the system has not violated any specified constraints.
6. **LONG TERM ASSURANCE**: To give some assurance that the long term characteristics of the system are as planned. Things like adaptability, maintainability.
7. **LEGAL COMPLIANCE**: make sure the system is always compliant with legal and other compliance policy items.

QA Strategies: valid QA Strategies include, but are never limited to:

1. **Clear and quantified project and system level objectives**, at the level of the organization, the product (example a software package), and the system (all related aspects such as support, documentation, marketing). Using 'Planguage'* fully to express them clearly and fully.
2. **Clear, detailed, impact and cost estimated strategies** for meeting the objectives. Rated and measured using Impact Estimation* specification language. Strong architecture design to meet multiple quantified objectives and constraints.
3. **Quality Control Reviews** of all forms of specification and documentation against sufficiently high standards (Rules, Processes, and other types of Standards * The reviews ("Spec QC" *) will operate at early stages (leading measures) to
a. **motivate** and **teach** engineers to use best practices in practice
b. **measure** the degree to which every development and maintenance output actually meets the best practice adopted standards.
c. give a **basis for** serious numeric process entry and exit conditions* for all engineering work outputs.
D. give a partial **data-driven basis for** continuous and immediate process improvement (like Defect Prevention Process (DPP) and CMMI Level 5).
4. **Rapid Evolutionary iteration**, incremental delivery, data collection, feedback, analysis and change (as in Evo *). Delivery being really useful value increments to stakeholders. The purposes of this are:
a. to be able to **prioritise** high value deliveries very early
b. to **validate** that the teams are actually able to **deliver value** at all
c. to **learn rapidly** about everything, and improve everything rapidly.
D. to enable all Quality Assurance tactics **to be tried and proven**, early and often.
5. **Automated and Built in** measurement and detection of problems, including very direct real-time user and stakeholder feedback to developers and engineers. Operating before releases and eternally after releases.
6. Ongoing Fact Analysis: **Software Engineering Accounting**: capability of analysing, organisation wide and in the long term how all these things are working, so as to determine what works best and what the costs are.
7. oh yes, I almost forgot, **conventional testing, at its best.**

* as detailed in Gilb: Competitive Engineering – or any better or equivalent specification method.

QA Principles (general ideas that guide us in detailed decisions)

1. EARLY BIRD: QA must operate **as early as possible** to detect problems both in the current development/maintenance process, and in the product or system being engineered.
2. **Quantification** will always be preferred as a language to express any variable idea associated with product, system and process ideas. It is always possible, and it is the only sound basis for rational thinking by management, engineering, and academic research.
3. **Prevention**: we will rapidly invest in a shift to prevention of problems, rather than tolerate eternal levels of problems to clean up.
4. **Rapid Feedback**: we will position our activities to get rapid feedback (like this same week) so we can correct bad things as soon as possible, and they cannot fester for months and years.

QA Values

1. **Efficiency**: we will always try to find and implement the most cost effective methods to assure system quality in the long term.
2. **Confidence**: our assertions of confidence in system releases can be relied on totally, and will be very explicitly about caveats, assumptions, maximum deviations, and responsibilities if accepted.
3. **Predictability**: we want to develop our ability to predict system attributes based on early indicators (example field bugs based on requirement major defects)
4. **Leanness:** we will constantly remove and avoid all activity that does not have clear measured value contributions in relation to its real cost.
5. **Perceptiveness**: we will anticipate all system quality aspects even when not directed by our stakeholder to explicitly deal with them.
6. **Reality**: we will be directed by current local realities: how things work for us now. We will not be driven by ideals, fads, customs, misunderstanding, illogical arguments.
7. **Delegation – avoiding micromanagement**: we will practice extreme delegation for details and choice of design, and for processes, to the people who do them daily. Management's role will be limited to setting

clear measurable high level objectives for products and processes, and then enabling their staff to reach them.

The IT Management role in Making Real QA Happen

The *testers* have not been leading the change to real and complete QA. They ‘test’.
The *developers* have not done anything laudable either. They code.
Managers don’t seem to have a clue, but then nobody actually trained them to understand broad QA.
The *universities* have no discernible honour in educating us in broad QA.
But, nothing is going to change unless responsible management (CIO, CTO, if necessary CEO levels) is somehow illuminated about QA, and chooses to insist it is exploited to its fullest potential.
So, this manifesto:

The following signatories **claim** to be able and willing to help management achieve the above manifesto elements, in *part* or whole.

And *they will on request, directly or publicly, give reasonable evidence of their real capability in terms of experience, results, facts, measures, references and writings (papers, books, slides) by or about their efforts to date; so that managers might fairly evaluate their potential value:*

Signatories:
Tom Gilb gilb.com 22 May 2009
Kai Gilb gilb.com 22 May 2009

manifesto : a public declaration of policy and aims, esp. one issued before an election by a political party or candidate.
ORIGIN mid 17th cent.: from Italian, from ***manifestare***, from Latin, ***‘make public,’*** from ***manifestus ‘obvious’*** (see **manifest** ^{1}).

False QA: is calling your activity ‘QA’ when in fact you only do testing.

References:

Gilb, Tom, ***Competitive Engineering***, A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage, ISBN 0750665076, **2005**, Publisher: Elsevier Butterworth-Heinemann. Sample chapters will be found at Gilb.com.