A Guide to the New IEEE 730 Software Quality Assurance Standard

David I. Heimann

Contents

- Introduction to IEEE 730 and SQA
- SQA Process Implementation
- Product Assurance
- Process Assurance
- Annexes and Summary

INTRODUCTION

What is IEEE 730?

- **Gives guidance and establishes requirements for Software Quality Assurance in a software project.**
- **The very first published software engineering standard – 1979.**
- **Gives the whole story for the Software Quality Assurance tasks outlined in IEEE 12207 -- Software Life Cycle Processes.**
- **IEEE 730-2014 greatly expands on the previous version of 2002; more like a whole new standard than a revision!**

Why use IEEE 730?

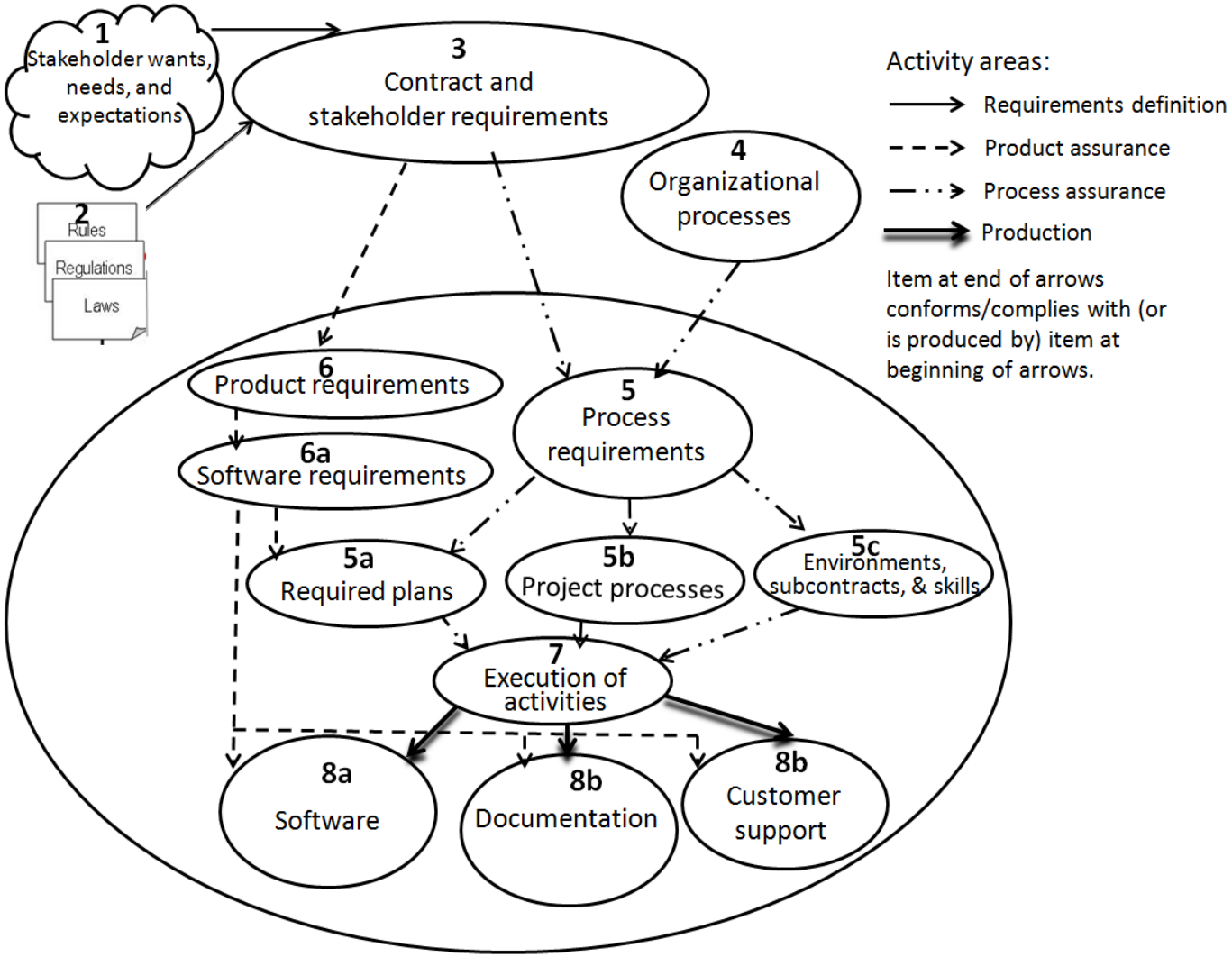
- Easy to use, very informative
  - Easy to follow, like a handbook
  - Gathers all the current SQA information in one place
  - Provides a clear checklist of what to do to organize the production of quality software
- Fulfills important quality purposes for an organization
  - Demonstrating conformance to the official standard for SQA
  - As a reference for developing an effective and consistent SQA process specifically pertinent to the organization
  - Obtaining information and guidance for specific questions

What Is Software Quality Assurance?

- A set of activities that →

  1. defines and assesses the adequacy of software processes to →
  2. provide evidence for a justified statement of confidence that →
  3. the software processes will produce software products that →
  4. conform to their established requirements.

SQA Links These Plans, Processes, and Products



Why SQA?

- Fewer defects in the processes used to develop software.
- Fewer defects in business rules and requirements.
- Fewer defects in the software products.
- Defects are found much earlier in lifecycle and so cost far less to address.
- Reduce and eliminate waste.
- Generate confidence throughout the lifecycle that activities will go well.

You Don’t Want This

http://www.amazingonly.com/cartoon/software-bugs-life/

SQA Is Not

- Testing
- Reviewing or Auditing
- Done only at the end of development
- Reactive
- A gate or "police"
- An organizational unit (though some units may be named "SQA")

SQA ACTIVITY AREAS

SQA Activity Areas

I. SQA Process Implementation

II. Product Assurance

III. Process Assurance

There are 16 SQA tasks in these 3 activity areas

Process Implementation Tasks

1. Establish the SQA Processes
2. Coordinate with related software processes
3. Plan SQA activities
4. Execute the SQA Plan
5. Manage SQA records
6. Evaluate organizational objectivity

Product Assurance Tasks

7. Evaluate plans for conformance
8. Evaluate products for conformance
9. Evaluate products for acceptability
10. Evaluate product lifecycle support for conformance
11. Measure products

Process Assurance Tasks

12. Evaluate **lifecycle processes** for conformance
13. Evaluate **environments** for conformance
14. Evaluate **subcontractor processes** for conformance
15. **Measure** processes
16. Assess staff **skill and knowledge**

PROCESS IMPLEMENTATION

I. Process Implementation

Dilbert, by Scott Adams, via http://madhusudhan.info/Comics/Dilbert/

Task 1 – Establish the SQA Process

Define an effective SQA process that identifies what to do and how to:

1. **Do it well**
2. **Confirm it is done right**
3. **Measure and track it**
4. **Manage and improve it**
5. **Encourage using it to improve quality**

Task 2 – Coordinate with Related Software Process

Enable SQA to integrate activities with other software processes, such as:

1. **Verification, Validation, Review, and Audit**
2. **Project Planning**
3. **Technical Processes**
4. **Implementation Processes**
5. **Reuse Processes**
6. **Agreement**

Task 3 – Planning the SQA Activities

- **Adapt the generic SQA processes to the specific needs of the project.**
- **Results are documented in the Software Quality Assurance Plan (SQAP).**
- **This is where SQA is adapted to the specific nature of the project (e.g., Agile, CMMI, embedded, etc.)**

Outline for the SQA Plan

1 Purpose and scope
2 Definitions and acronyms
3 Reference documents
4 SQA plan overview
    4.1 Organization and independence
    4.2 Software product risk
    4.3 Tools, techniques, and methods
    4.4 Standards, practices, and conventions
    4.5 Effort, resources, and schedule
5 Tasks, activities, and outcomes
    5.1 Product assurance
    5.2 Process assurance
6 Additional processes
    6.1 Contract review
    6.2 Quality measurement
    6.3 Waivers and deviations
    6.4 Task repetition
    6.5 Risks to performing SQA
    6.6 Communications strategy
7 SQA records
    7.1 Analyze, identify, collect, file, maintain and dispose
    7.2 Availability of records

Task 4 – Executing the SQA Plan

- **Execute the SQAP.**
- **Revise the SQAP as appropriate.**
- **Raise non-comformances when products or processes do not conform to their requirements.**
- **Create and use SQA records to improve quality.**

Task 5 – Manage SQA Records

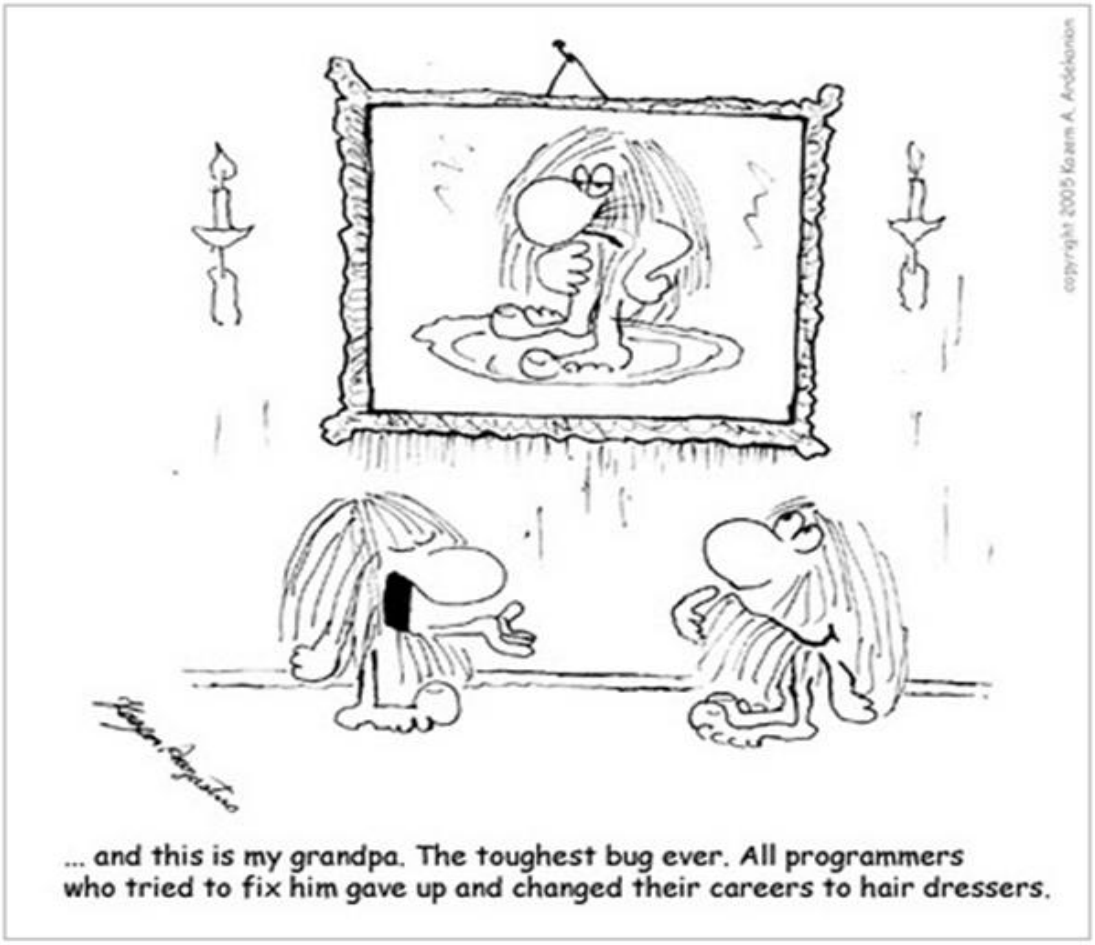
- **Records are created, maintained, and made available to project personnel and management.**
- **Records aim to document that project activities:**
  - Are performed in accordance with project plans.
  - Comply with the contract.
  - Support the identification and rectification of problems, causes, and improvements.
  - Enable information sharing.

Task 6 – Evaluate Organizational Objectivity

- **Those who perform SQA activities must have the organizational objectivity and authority to make objective evaluations and verify problem resolutions.**
- **Three important aspects of objectivity are:**
  - Technical Independence: Not involved in the development of the products being evaluated.
  - Managerial Independence: Not reporting to individuals responsible for product development/project management.
  - Financial Independence: Budget not controlled by individuals responsible for product development/project management.

PRODUCT ASSURANCE

II. Product Assurance



http://www.amazingonly.com/cartoon/software-bugs-life/

Product & Process Assurance

- **Product Assurance**
  - ✓ **Software products conform to established requirements**

Task 7 – Evaluate Plans for Conformance

1. Identify plans required by the contract.
2. Raise non-conformances when plans do not conform to the contract (or when the contractural requirements are inadequate).
3. Raise non-conformances when plans are not mutually consistent.

Task 8 – Evaluate Products for Conformance

1. Identify products/documentation required by the contract.

2. Identify allocated requirements and ensure adequacy.

3. Ensure that evaluations of software products/documentation for conformance against the requirements are performed.

Task 9 – Evaluate Product for Acceptability

- **Determine project's understanding of conditions for product acceptance.**
- **Prior to delivery, evaluate the level of confidence that the software products and related documentation will be acceptable to the acquirer.**

Note -- Depending on contractual agreements (e.g., Agile environments), the customers themselves may make some acceptability determinations prior to delivery.

Task 10 – Evaluate Product Support

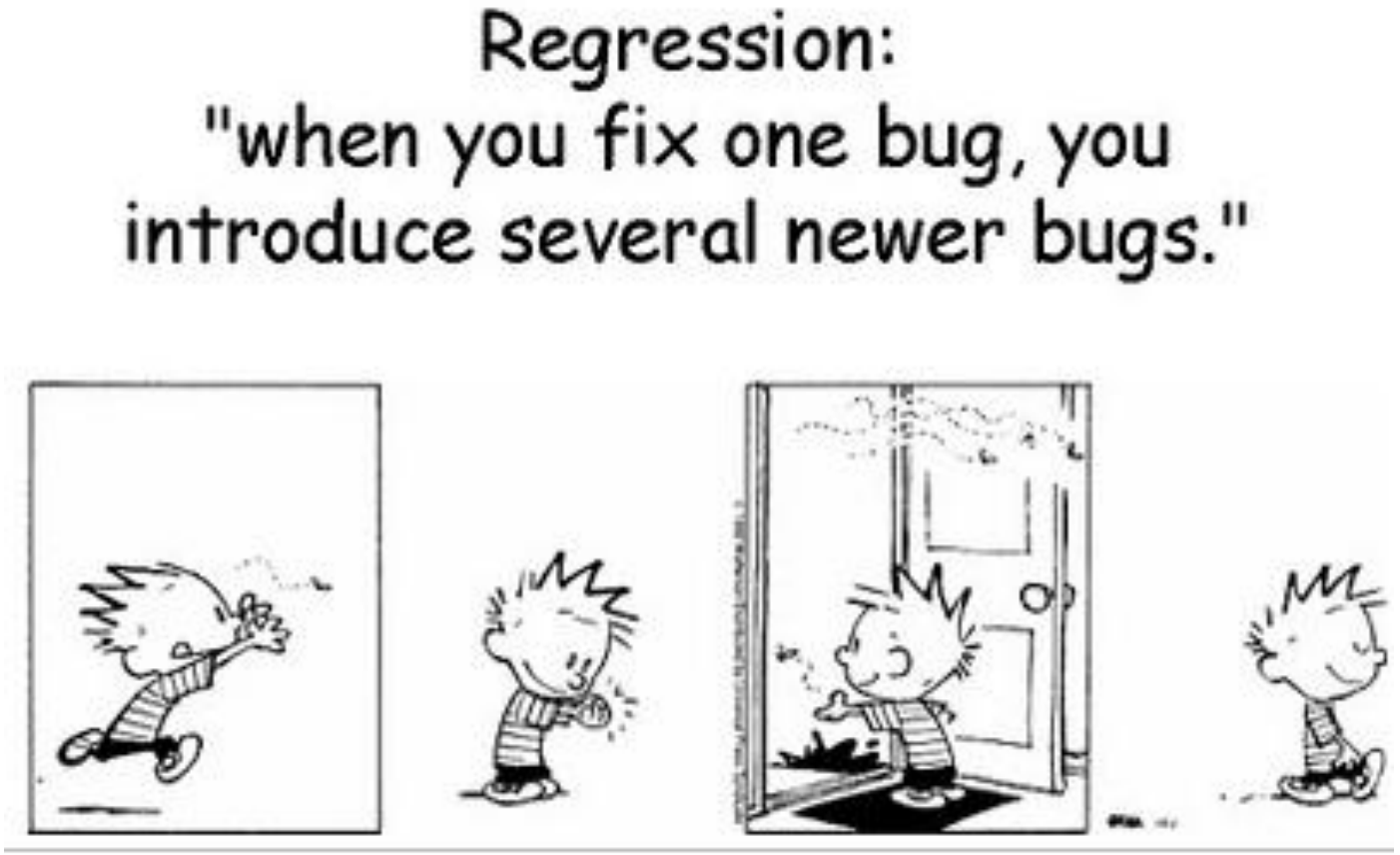
- **Have acquirer's expectations for product support and cooperation been established and documented?**
- **Have they been met?**
- **If the SQA process ends at delivery, how is suitable support ensured?**

Task 11 – Measure Products

- Do the project measures accurately and objectively represent the quality of the software products?
- Are improvements done as a result of the product measurements effective in improving product quality?
- Do the measurements of software products satisfy the measurement requirements and conform to the measurement plans?

PROCESS ASSURANCE

III. Process Assurance



http://softwaretestingandqa.blogspot.com/ (and Calvin & Hobbes)

Process Assurance

- **Process Assurance**
  - ✓ **Project activities conform to accurate and effective defined processes**

Task 12 – Evaluate Life Cycle Processes

- Does the software development life cycle conform to project plans and fit with contractural requirements?
- Does the execution of project activities conform to the project plans?
- Does the execution of project activities yield products that conform to requirements?

Task 13 – Evaluate Environments

- Do the software development environments conform to project plans?
- Do the software test environments conform to project plans?

Task 14 – Evaluate Subcontractor Processes

- **Do subcontractor processes conform to requirements passed down?**
- **Have acquisition needs, goals, product, and service criteria been identified? Have they been met?**

Task 15 – Measure Processes

- Do the project **measures** support effective management of the **software processes**?
- Do the project measures meet the **information needs** necessary for managing effective processes?
- Does the executed measurement process satisfy the measurement requirements and **conform** to the measurement plans?

Task 16 – Assess Staff Skill & Knowledge

- **Do the staff, including SQA staff, assigned to the project have the knowledge, skills, and abilities to perform their assigned roles?**
- **Have education and training plans been developed? Are they effective?**

ANNEXES AND SUMMARY

Annexes

A. Mapping between IEEE 12207 and IEEE 730
B. Mapping between SQA Plan outlines in IEEE 730-2002 and IEEE 730-2104
C. Guidance for Creating Software Quality Assurance Plans
D. Mapping between IEEE 730 and SPICE
E. Industry-Specific Guidance for IEEE 730
F. IEEE 730 and the Agile Development Process

Annexes

G. IEEE 730 and Very Small Entities (Std 29110)
H. Software Tool Validation
I. Assessing Product Risk: Software Integrity Levels and Assurance Cases
J. Corrective and Proventive Action Processes and Root Cause Anallysis Process
K. Cross-reference
L. Bibliography

IEEE 730 and Agile

- In Agile, the product backlog plays a role of the "contract". 730 shows how to use the product backlog in its role as a contract.
- The product SQA portion of SQA Plan specifies the Agile "done" criteria.
- Non-conformances are inserted into the backlog and addressed in the appropriate sprints.
- Evaluation of product for acceptance is a continual process in Agile, not just at end of project.
- IEEE 730 has an annex on Agile with further details.

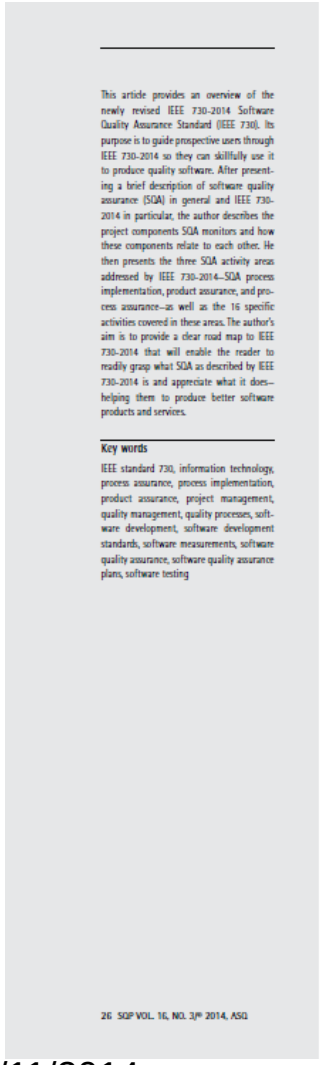
IEEE 730 and CMMI

- CMMI has 16 core process areas. The two that relate to quality are PPQA (Product and Process Quality Assurance) and VER (Verification).
- Since CMMI does not specify a particular process flow, CMMI-conforming organizations need to design their own PPQA process.
- IEEE 730 provides details for this process design.
- VER process area implements product quality assurance according to the plan in PPQA. 730 covers both product and process quality assurance.
- 730 has associated materials with maps between 730 and CMMI.

Summary

- IEEE 730 provides a foundation for Software Quality Assurance, which in turns provides confidence that software products will conform to their established requirements and satisfy the customer.
- IEEE 730 addresses the three areas of SQA: Process Implementation, Product Assurance, and Process Assurance.
- IEEE 730 can be used to prove conformance where SQA conformance is required, and to provide guidance where SQA conformance is desired.

Available Articles (see sign-up sheet)



This article provides an overview of the newly revised IEEE 730-2014 Software Quality Assurance Standard (IEEE 730). Its purpose is to guide prospective users through IEEE 730-2014 so they can skillfully use it to produce quality software. After presenting a brief description of software quality assurance (SQA) in general and IEEE 730-2014 in particular, the author describes the project components SQA monitors and how these components relate to each other. He then presents the three SQA activity areas addressed by IEEE 730-2014—SQA process implementation, product assurance, and process assurance—as well as the 16 specific activities covered in these areas. The author's aim is to provide a clear road map to IEEE 730-2014 that will enable the reader to readily grasp what SQA as described by IEEE 730-2014 is and appreciate what it does—helping them to produce better software products and services.

Key words

IEEE standard 730, information technology, process assurance, process implementation, product assurance, project management, quality management, quality processes, software development, software development standards, software measurements, software quality assurance, software quality assurance plans, software testing

GENERAL KNOWLEDGE

An Introduction to the New IEEE 730 Standard on Software Quality Assurance

DAVID I. HEIMANN

INTRODUCTION

Software quality is defined as how well the software meets its established requirements and stakeholder wants, needs, and expectations. It is one of the key attributes (along with functionality, quick time to market, reasonable cost, reliability, and safety) that software must have to be a successful product and a source of profit to the organization developing it. To assist organizations in producing quality software, the document "IEEE Standard for Software Quality Assurance Plans" was developed in 1981 under the standard number 730 and has been updated periodically since.

The Institute of Electrical and Electronics Engineers (IEEE) has revised and upgraded this standard, last changed in 2002, through a technical working group that includes the author of this article. The revised standard, whose title has changed to "IEEE Standard for Software Quality Assurance Processes" (IEEE 2014) and is referred to in this paper as "IEEE 730-2014," has been approved and will be released later this year (2014). However, it should be noted that the final version of the standard may include additional changes not known at the time of this publication.

With its expanded and detailed coverage, informative annexes, and improved readability, the new standard goes beyond merely addressing the development of software quality assurance

SQA Activities and Their Relationship to the Agile Development Process

1. Introduction

Agile methods, such as Scrum, Extreme Programming, Dynamic Systems Development Methodology (DSDM), Adaptive Software Development, Lean Methodology, and Feature Driven Development (FDD) are approaches to building software to adapt to rapidly changing customer requirements. These methods provide software suppliers the ability to respond in an agile manner.

These approaches typically follow the twelve principles below (See www.agilemanifesto.org/principles.html for more information.)

1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2) Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4) Business people and developers work together daily throughout the project.
5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7) Working software is the primary measure of progress.
8) Agile processes promote sustainable development. The sponsors, developers, and users are expected to be able to maintain a constant pace indefinitely.
9) Continuous attention to technical excellence and good design enhances agility.
10) Simplicity—the art of maximizing the amount of work not done—is essential.
11) The best architectures, requirements, and designs emerge from self-organizing teams.
12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile approaches include but are not limited to the following elements:

- Burndown charts
- Collaborative development
- Collective code ownership
- Continuous feedback
- Continuous integration
- Customer involvement
- Pair programming
- Refactoring
- Small development teams
- Small releases
- Sprint/Timeboxes
- Test-driven development

David I. Heimann, Ph.D.

E-mail: heimann.david@gmail.com

Phone: 617-524-4531