# 10 Proven Principles for Process Improvement and Organizational Success

SEPG Conference 2007

Dr. Richard Bechtold
Abridge Technology: www.abridge-tech.com

# 10 Proven Principles for Process Improvement and Organizational Success

1. Avoid Fixing the Invisible
2. Throw Away Cautiously
3. Target Senior Experienced Professionals
4. Model Twice, Define Once
5. Aggressively Avoid Perfection
6. Encourage Complaints
7. Accelerate History
8. Resynchronize Later
9. Leverage Resistance as a Source of Next Steps
10. Ensure the QA Team Does Nothing

## 1. Avoid Fixing the Invisible

- Process improvement is difficult enough when you can see what you are doing
- Attempting to improve an invisible process is nearly impossible
- Two primary techniques for achieving or enhancing process visibility are
  - Process models
  - Process descriptions.
- In the absence of either models or descriptions, process improvement often consists of ongoing debates by people who usually have dramatically different mental models

## 1. Avoid Fixing the Invisible

- An early and critical step in successful process improvement is the rapid development of
  - Very brief, unambiguous policy statements
  - A few simple process models
  - Sparse process descriptions
- At this stage you literally only need enough to see what you are actually talking about
- Enhancements can come later

## 2. Throw Away Cautiously

- Process improvement efforts don't always work as planned
- Sometimes on the failed efforts—and especially on the major failures—there is a strong tendency to
  - Throw everything away
  - Start all over again
- This is almost never a good idea.

## 2. Throw Away Cautiously

- Implicitly, throwing everything away means that there was absolutely nothing of value worth saving
- In fact, there's almost always material worth retaining and updating—maybe more material, or maybe less, but it invariably exists
- Always build on the work that was done before your arrival

## 3. Target Senior Experienced Professionals

- Once you assign someone to develop a process description, one of the first questions they’ll ask is, “How much detail do you want?”
- It is impossible to answer that question without considering the intended audience for your process descriptions
- Initially, make your target audience senior experienced professionals

## 3. Target Senior Experienced Professionals

- Generally, avoid detailed descriptive tutorials
- Strive instead for more of a “reference material” style where
  - Answers to questions can be quickly located
  - Pertinent information can be easily understood
- Err on the side of “too little” and “too sparse”—if people actually want more information, let them ask for it

## 4. Model Twice, Define Once

- In the field of carpentry there is an expression, “Measure twice; cut once”
- In the field of process improvement there is a similar relationship between process models and process descriptions
- Many organizations mistakenly attempt to develop process descriptions in the absence of process models
- This is almost exactly analogous to writing software code without putting any effort into developing the software design

## 4. Model Twice, Define Once

- To minimize overall costs, make the relatively minor investment to design or model your processes as an early priority
- Do this well before you commence the more substantial investment of creating or making major updates to actual process descriptions
- Distribute the models for review and feedback so you can find and fix the flaws early and inexpensively

## Model Twice, Define Once

- Be very careful in your selection and use of a modeling tool
  - If a tool makes it very easy to create really complex diagrams, what do you think is going to happen?
- The most important objective of a model is to clearly communicate *to the intended model user*

## 5. Aggressively Avoid Perfection

- In the world of process improvement, striving for perfection is a disaster
- The basic truth is you'll never get it perfect, regardless of how much money, time, and effort you pour into it
- For example, when developing plans sometimes you just know the plan is probably wrong due to
  - Key information you don't yet have
  - Kay factors (and future events) you are not aware of

## 5. Aggressively Avoid Perfection

- One option is to wait until you know everything (but this typically won't happen until well after the project is completed)
- The other option is simply to ask yourself, is this initial version of the plan barely adequate?
- If the answer is yes, then go ahead and release it as version 1.0.
- By definition, if it's adequate, it's usable
- Use the barely adequate version to get started and, as with everything else, improve it over time

## 6. Encourage Complaints

- As you pilot and deploy processes, you really want to hear from practitioners regarding anything whatsoever that
  - Doesn’t appear to be working
  - They simply don’t like
- In particular, you are looking to conduct “demand-driven” process improvement

## 6. Encourage Complaints

- For example, if you deployed sparse process documentation some projects may tell you it is just too abstract
- When people declare a new process is not working, ask them, “What’s preventing it from working?” (Missing templates, no examples, training wasn’t received, etc.)
- If effect, people who are complaining are often just showing you where new or supplemental material is needed

## 7. Accelerate History

- A common challenge in system and software environments is gathering performance data for
  - Process capability baselining
  - Project estimation
  - Understanding (statistically) performance and product variations
- Ideally, we have historical data, but frequently historical data doesn't exist
- In that context, develop "rough order of magnitude" estimates regarding performance, duration, etc. (which is what you were likely going to do anyway)
- Then, adopt a highly incremental development lifecycle where the phases are designed to be similar to each other

## 7. Accelerate History

- Collect data relating to actual performance during the first several phases
  - Example: two-week phases, conducted over the three month period
- Strive to keep each phase reasonably similar so you can compare data from various phases
  - Find methods for “factoring out” or adjusting for unusual influences on the data
- You will soon have highly applicable “historical” data
- By systematically collecting performance and quality data you can conduct ongoing refinements to your cost, schedule, quality, variance, and other models

## 8. Resynchronize Later

- In principle, requirements drive the design, and design drives the code
- This works great early in a project, but frequently doesn't work at all during the later phases of a project
- For example, if you've shipped systems to key customers for beta evaluation, and you receive a report of a devastating bug, you typically need to immediately
  - Research the cause
  - Fix the code
  - Retest
  - Ship the update

## 8. Resynchronize Later

- In this scenario, nobody wants to wait while you
  - Update the design documents
  - Update the requirements specifications
- You primary objective is to get the fix out the door
- This is typically not only normal but it also reflects a correct sense of priorities
- However, at some future date you do need to revisit the requirements and design specifications (and other constraining artifacts) and ensure they are updated
- Routinely conduct periodic resynchronizations of source documents with the latest product releases—including all the emergency updates you've been allowing

## 9. Leverage Resistance as a Source of Next Steps

- As a rule, people resist your attempted process improvements for a reason
- Sometimes, it's just reluctance to change
- However, as the expression goes, "Sometimes resistance is just process improvement in disguise"
- When you encounter resistance, always carefully investigate whether or not those resisting are doing so for a very good reason

## 9. Leverage Resistance as a Source of Next Steps

- Additionally resistance typically comes in two categories
  - Active, overt, obvious, and confrontational
  - Passive, stealthy, subversive, and disguised as support
- Which of the above do you want to encourage?

# 10. Ensure the QA Team Does Nothing

- An excellent strategy for implementing your quality assurance function (QA) is to ensure that the quality assurance team does not actually "do" any lifecycle phases, nor any other functions
- Instead, they should focus 100% on double-checking what other people are doing, e.g.,
  - Testing is done according to plan
  - Progress is being made with regard to CMMI compliance
  - Peer reviews are conducted on schedule
  - Required records are being kept by others

## 10. Ensure the QA Team Does Nothing

- Many organizations have placed software testing under the quality assurance function
- Although a very common practice, this creates a significant problem: you no longer have anyone who can objectively double-check that testing is being performed as required
  - The quality assurance team cannot objectively perform quality assurance on the testing activities that their own organization is performing

## 10. Ensure the QA Team Does Nothing

- Hence, an excellent strategy is to always use quality assurance as a safety net
- Let the engineering, management and functional area performers do whatever spectacular performances are required
- Have quality assurance as the safety net for whenever anyone accidentally falls
- And remember, it's never a good idea for you to distract your safety team

## 10 Proven Principles for Process Improvement and Organizational Success—Summary

1. Avoid Fixing the Invisible
2. Throw Away Cautiously
3. Target Senior Experienced Professionals
4. Model Twice, Define Once
5. Aggressively Avoid Perfection
6. Encourage Complaints
7. Accelerate History
8. Resynchronize Later
9. Leverage Resistance as a Source of Next Steps
10. Ensure the QA Team Does Nothing

## Biographical Highlights

Dr. Bechtold is a senior consultant for Abridge Technology, a Virginia-based company he founded in 1996. Dr. Bechtold provides consulting, training, and support services in the areas of project management, process improvement, process definition, measurement and statistical techniques, and risk management. Dr. Bechtold has assisted government and industry with implementing the Software CMM since 1992, the Acquisition CMM since 1996, and the CMMI since 2000. Dr. Bechtold's expertise spans organizations of all types and sizes. Abridge Technology is an SEI Partner, and Dr. Bechtold is an authorized instructor of the SEI's, “Introduction to CMMI”.

## Contact Information

Dr. Richard Bechtold

President; Senior Consultant

Abridge Technology; Ashburn VA

703.729.6085

rbechtold@abridge-tech.com

www.abridge-tech.com