

Team Retrospectives — for better iterative assessment

by [Ellen Gottesdiener](#)

Principal
EBG Consulting

To build a positive project team it is essential to make self-reflection and continuous learning a part of your organization's culture. Perhaps the single most efficient way to do this is by incorporating ongoing team retrospectives into your project. You can think of these retrospectives as the "flesh" of the iteration and milestone assessments that the Rational Unified Process,® or RUP® recommends. The RUP provides only a skeleton for these assessments, leaving teams to structure the activities on their own. RUP suggests that project team members gather after each iteration and milestone to reflect on their performance and process. A team retrospective approach gives structure to these sessions. The purpose is to provide progressive learning, sustenance, and improvement. A hallmark of agile teams, retrospectives allow all team members to iteratively test and develop not only solid software but also effective teamwork



Do Your Assessments Need More Structure?

Somewhere in a corner conference room, a team is holding a meeting that should be an iteration assessment, but they are instead focusing on what to do next. If we were magically to become privy to the thoughts behind their words, here's how the discussion would go...

- ▶ [subscribe](#)
- ▶ [contact us](#)
- ▶ [submit an article](#)
- ▶ [rational.com](#)
- ▶ [issue contents](#)
- ▶ [archives](#)
- ▶ [mission statement](#)
- ▶ [editorial staff](#)

What they say:	What they are really thinking:
<p>Tom (project lead): <i>"Okay, now that we're finished with the first iteration, we'd better get going and add these three use cases for iteration two."</i></p> <p>Rita (requirements analyst): "I admire your optimism, Tom."</p>	<p><i>"Yikes! If we don't catch up, we're going to really get behind!"</i></p> <p><i>"How can we start the next iteration? We haven't finished the prototype with the two additional scenarios the customers added on Tuesday. They blew their stack when they didn't see those scenarios yesterday! What am I gonna tell them?"</i></p>
<p>Tom: "Thanks, Rita. Joan, the test team needs to make those scripts available quicker. Can you do that?"</p> <p>Joan (test lead): "We'll try, but we may be short-handed this week."</p>	<p><i>"Gotta cut a week and a half somewhere, or this project's gonna be toast."</i></p> <p><i>"Oh, give me a break! They didn't define the requirements or scenarios until after they started to code! So now they expect my team to take up the slack. How can we do that?"</i></p>

Seem familiar? If so, it may be comforting to know that your team is not unique. Unfortunately, even teams that have adopted many of the best practices in RUP often do not understand the purpose of an iteration assessment. They attend the meetings but focus only on what's ahead instead of looking back over what they've just completed. They don't share their conflicting concerns and needs or their differing perspectives. As a result, they move from iteration to iteration or project to project, repeating the same mistakes and failing to capitalize on their successes. The team never achieves its potential in terms of both project results and building a knowledge base for both present and future team members.

Fortunately, there is a simple and inexpensive way to turn things around -- to help team members learn how to stop repeating mistakes and re-creating success by accident. It's called structuring your assessments as a team retrospective.

What Is a Team Retrospective?

Unlike a classic meeting, which typically involves sharing information or status, a retrospective is a tool for learning that also generates new

information and action plans. It is a means of assessing not only the condition of the code and technology, but also the team's performance and the quality of their teamwork. The team uses a skilled, neutral facilitator to guide its work, and includes everyone who was involved in creating the deliverables the group is reviewing or was otherwise involved in the project during the same time period.

Retrospectives can be incorporated into iteration and milestone assessments as well as post-project reviews. For a typical iteration or milestone assessment, a retrospective will take two to six hours; you will need more time if you are retrospecting a completed project.¹

Iterating Requires Continual Retrospection

An iterative development lifecycle such as the one outlined in the RUP is composed of a series of cycles (iterations). The team plans work, does it, checks it, takes corrective action or ensures they repeat prior desirable behaviors, and moves on. They repeat this cycle -- essentially the "plan-do-check-act" process known in the quality improvement community -- across all phases (Inception, Elaboration, Construction, and Transition) until they deliver the entire product (see Figure 1). In an iterative development lifecycle, the project is treated as a live learning lab; project quality evolves through the ongoing, corrective feedback gained from each iteration via an iteration assessment, and from each phase via a milestone assessment, both of which can be structured as retrospectives. As we'll discuss, retrospectives offer other benefits, but their main job is to give you a structure for exploiting the advantages of an iterative development cycle.

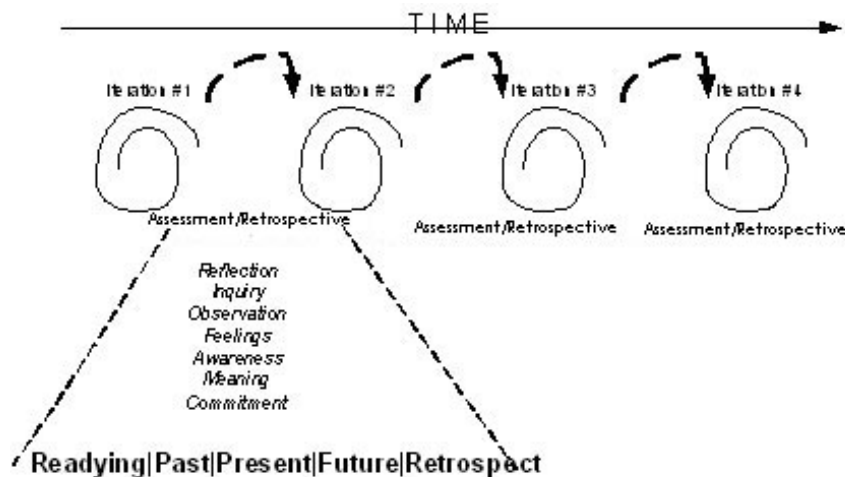


Figure 1: Assessments/Retrospectives for Iterative Projects. Each project includes an assessment/retrospective following each iteration. The retrospective includes a series of steps: Readyng, Past, Present, Future, and Retrospect the Retrospective. The team's job is to reflect, inquire, analyze, and use what it learned to plan the next iteration.

Learning from Retrospectives

A retrospective is a reality-based learning experience resulting in action

and change.

For learning to stay with you, it must have immediacy, relevance, and self-direction. Let's translate that into implications for your team retrospectives.

- *Immediacy* means that you have opportunities to apply what you learn very soon after the learning occurs (the thought process is, "I can remember it"). That's why it's important to time your retrospectives within days of the endpoint you are retrospecting.
- *Relevance* means that the learning is important to us, and that it applies to our current situation ("I care," "I need it," and "I can practice or test this -- real soon"). You should plan your retrospective around the team members' goals for the session, being sure to cover topics the team cares about.
- *Self-direction* involves taking ownership and control of our own learning and making necessary changes ("I own it," "I choose it," and "I will do it"). Always include steps in the retrospective for everyone to decide and plan what to change and how to change it.

Planning and Running Successful Retrospectives

Successful retrospectives have a number of characteristics:

- They are planned.
- They are held multiple times throughout the project.
- They involve the project community.
- They are led by a neutral, skilled facilitator.
- They use data from the project.
- They acknowledge that feelings count.
- They follow a structure.
- They are the basis for change.

Let's look at each of these characteristics in detail.

Plan for a Successful Retrospective

In advance of the session, the team, with the help of the retrospective facilitator, plans what will be covered, who will be there, and how long the retrospective will take. For example, a team retrospecting on the completion of its Inception phase might reflect on customer involvement, management support, and team communications. A team retrospecting on the first iteration of Construction might focus on development tool effectiveness, defect detection, and architectural design. Here are some questions the team might cover:

- How clear was our product vision when we began the iteration? How

clear is it now that we have finished it?

- Were customers involved appropriately in this iteration? How did they react to the work we did?
- Based on this iteration, does our project plan seem sound, or does it need changing?
- What risks did we reduce in this iteration? What risks do we still face, and do we have effective ways to manage them?
- How well did the team communicate during this iteration? If there were misunderstandings or failures, why did they happen, and how can we make improvements?
- Did we have the right people to cover essential roles and responsibilities? Were the lines of responsibility clear to all team members? Did our team structure and organization work effectively?
- What tools and technologies did we use in this iteration? How effective were they?
- How well did the decisionmaking process work in this iteration? If anything went wrong, why did it happen?
- Did management provide adequate support for our work? If not, what could they have done differently?
- Are we working with high-quality requirements? How volatile are the requirements, and why?
- Do we have an effective change control process to guard against scope creep? How well did we manage scope during this iteration?
- How effective was our test plan? What did the tests we conducted point out?
- Are we creating or working with a high-quality architecture?
- Are we generating end-user documentation and training materials as we go and testing them on end users?
- Are our customers getting ready to work effectively with the new product? Did they make organizational or culture changes during this iteration to prepare for implementation?

Align the content of your retrospective with the goals for the iteration you've just completed. For example, suppose you have finished an iteration of the Elaboration phase with the goals of verifying the breadth of requirements, testing the validity of the overall project scope, and finding missing requirements. Your post-iteration retrospective should focus on requirements quality and volatility.

Table 1 provides a list of example questions your team might address in retrospectives for each phase of the RUP. (Also see an expanded list under **Retrospective Questions by RUP Project Phase** in the [Appendix](#).)

Table 1: Example Retrospective Questions by Project Phase

Project Phase	Potential Retrospective Questions
Inception	<ul style="list-style-type: none"> • How well did we delineate and communicate the product vision? • How clear is our scope? How might we make it more clear, if necessary? • How complete and accurate was our business case? • Are we using risks to guide the project? Why? Why not?
Elaboration	<ul style="list-style-type: none"> • Did we select the right strategy for creating our architectural prototype? If so, how did we know? If not, how did we know? • How much did our requirements change, and why? • How did we involve customers? In what ways did that involvement work to our benefit or work against it? • Was our architectural prototype sufficient to reduce risk and verify customer expectations? Is so, why? If not, why not?
Construction	<ul style="list-style-type: none"> • What was the quality of our code? How do we know? • How well did we plan and execute our testing? • How did our development, testing, and configuration management tools and processes work for us? What one thing would you change? • How did the testers and developers interact?
Transition	<ul style="list-style-type: none"> • How well did we make use of beta testing to stabilize our product? • How smooth was our plan and execution of database conversions? • Was the customer ready for the product? How do we know? • How effective is the product's support documentation?

End-of-Project	<ul style="list-style-type: none"> • What are you most proud of in this project? • What one moment stands out the most for you? • What did we learn about iterative development from this project? • What one recommendation would you make to other teams about their use of RUP?
All Phases	<ul style="list-style-type: none"> • What happened that surprised us? • How well are we communicating with each other? • Are we collecting the right metrics? • What one thing do you want to remember to do again in the next iteration?

Retrospect Multiple Times Throughout the Project

Structure every RUP review and assessment as a retrospective as you progress through the project. Perform a final assessment when you close out the project -- even if the project was canceled or postponed. Use feedback from each retrospective immediately to improve and plan your next set of project work.

It's a good idea to calibrate the timing and length of your retrospective to the length and importance of the iteration. If you've just completed a short, two-week iteration that involves a build using a new technology, your retrospective can be short and highly focused on the goals of the iteration. Remember that retrospectives can be as short as one or two hours, especially if the team is in a groove with the process and observes healthy team norms. (For more information, see **Defining Ground Rules for a Retrospective** in the [Appendix](#).)

Schedule iteration assessments/retrospectives for a few days after the end of the iteration. If you wait too long, memories of events will fade, or, even worse, team members will move on to the next iteration, and you'll miss out on planning for necessary changes and adjustments. Conduct your session in a comfortable space with blank walls for posting the work you'll do in the retrospective rituals. Make the team's collective knowledge visible, and you'll find it easier to examine the team's retrospective work. (For information about using walls for group work, see my article, "[Specifying Requirements with the Wall of Wonder](#)," in the November 2001 issue of *The Rational Edge*.) In addition, be sure to provide food or snacks for everyone during the session. If budgets are tight, ask different team members to make or bring food to each retrospective.

Involve the Project Community

Everyone who played a role in producing artifacts during the target period is part of your project community and should participate in the retrospective. Include analysts, testers, the project manager, and customers if appropriate. Because roles change, specific attendees may vary.

In one project I worked on, during Elaboration we did a retrospective for our third and final requirements iteration. The business analyst, customer representatives, the requirements analyst, lead architects, the project manager, and the test lead attended. Six weeks later, when we did an iteration assessment for a Construction phase iteration, the retrospective included the developers, the test team, the project manager, the architects, and customer representatives. Remember to include customers; they are integral to the software development process.

For a large project, you may ask only a few representatives from each project role to attend the iteration assessment/retrospective to keep the size manageable (you will need more than one facilitator for more than fifteen participants). These representatives must be willing to prepare by interviewing people in the community ahead of time to gather data concerning key questions you want to address in the retrospective. They must act as ambassadors for the larger role community they represent. They are also responsible for sharing outcomes with their representative community immediately after the meeting. The retrospective facilitator (see below) should also plan other ways to give those not present a voice.

Use a Neutral, Skilled Facilitator

Someone who is substantially neutral and viewed as such by the participants should facilitate your session. This person must have good facilitation skills, including the ability to manage the group when things get emotional, plan the retrospective, and select appropriate activities for the time allotted. The facilitator must be comfortable with helping the group address not only its performance but also its process. This might include addressing previously taboo issues, directly addressing conflict, and helping group members explore their inferences and assumptions about each other.

A team member with solid skills can facilitate as long as she believes she can be neutral and the entire group agrees. If your organization is big enough, it's a good idea to grow facilitation skills across various development and engineering groups and then swap them around for various group sessions, such as retrospectives and requirements workshops.

On one project I served as both a team member and the interim retrospective facilitator; when we set up session ground rules, I made team members promise to inform me if I wasn't managing to remain neutral. We were able to proceed with that understanding. I also moved to one specific place in the room when I wanted to make a content contribution to the discussion, and then moved back to the center of the room to resume my facilitator role.

Use Project Data

The team should collect and bring data to the iteration assessment to perform the retrospective for the prior iteration and to plan the next one.

Choose a small set of metrics that are useful for both purposes. One useful technique is the goal-question-metric (GQM) approach²:

- Define your goals.
- Ask relevant questions to check whether you achieved the goals.
- Derive metrics from the responses.

For example, if goals for an Elaboration iteration include saving time by using an optimum number of scenarios and increasing customer satisfaction by using prototypes effectively, you might ask questions such as:

- How many use cases were included?
- How many scenarios were generated and tested in the prototype?
- How many prototypes were developed and reviewed by the customer?
- How many days were needed to rework the prototype after a customer review?

(For more discussion on this topic, see http://www.processimpact.com/articles/metrics_primer.pdf.) This data should be captured so you can review it for patterns and trends and leverage it in planning future iterations and projects.

Project data can be useful to inform your inquiry into both team performance and process. You can ask "what if" questions to explore how your outcomes might have changed. For example, in one retrospective for an iteration assessment, we discussed the number of scenarios we generated as well as the ratio of scenarios to use cases, the number of business rules associated with each use case, and the number of workshops and customer reviews that we had used in that iteration. The data revealed a pattern of how many scenarios ("happy paths" as well as "unhappy paths," or exceptions) we used to elicit each use case and its associated business rules and to build a prototype. We speculated about how the iteration might have gone if we had used fewer use cases or more scenarios, fewer happy path scenarios, or more scenarios and fewer use cases. We also toyed with various ways to involve different customers to speed up the process. As a result of this inquiry, we adjusted our plans for defining and prototyping in our next iteration, and we delivered on our iteration goals with no major changes to the iteration prototype and very satisfied customers.

Acknowledge That Feelings Count

Because a retrospective goes beyond simply checking the outcomes or performance of a team, you should openly acknowledge the feelings of team members. Feelings may range from regret, sorrow, anger, and frustration, to pride, joy, and appreciation. When feelings are running strong, it can get in the way of getting the job done or interacting effectively, because we may selectively filter out information. A retrospective is not a therapy session, but it should not neglect feelings. A skilled facilitator knows that helping a team express feelings promotes healthy team collaboration. They do this by probing into the thinking and evidence that led to the feelings.

For example, during a retrospective for a Construction phase iteration, one developer replied curtly to comments that a tester was making. As facilitator, I noticed that the rest of the team immediately clammed up, and some people pursed their lips as if to say, "There he goes again!"

I restated what the developer had said and pointed out that his tone sounded angry; then I asked him whether I had interpreted his tone correctly. He sighed deeply and then admitted he was tired from working long hours and frustrated that his modules were the source of most of the outstanding defects. His anger turned to embarrassment and even shame; his anger was really directed at himself. Knowing this, the other participants were able to get past his "negative attitude" and see the real problem. The team began to explore what led to that situation and how they might help. This brief exchange helped improve the team's dynamics throughout the rest of the project.

A retrospective shouldn't focus solely on difficult, negative issues, although sometimes it is hard to elicit positive, complimentary feedback. Many organizational cultures value problem analysis and problem solving. But focusing on that to the exclusion of examining the half-full glass -- what is working -- involves a serious risk: that you won't be able to repeat your successes.

For that reason, retrospectives should incorporate appreciative inquiry -- the practice of seeking the positive core of the project's results and experience. You can include an appreciation ritual as the first step of a "Temperature Reading" (see **Example Retrospective Rituals** in the [Appendix](#) and Norm Kerth, *Project Retrospectives: A Handbook for Team Reviews*. Dorset House, 2001).

You can also use positive questions to explore topics that the team needs to address. For example, if the topic is communications, your facilitator can ask participants to recall a moment in the iteration when communication allowed them and another person to really connect and work exceptionally well together. You might consider:

- What were the circumstances?
- What made that communication compelling?

Next, the facilitator would ask participants to examine the present team and consider the various ways they communicate. Questions might

include:

- Which ways are most effective?
- Which foster a sense of connection and alignment with our project goals?
- Which enable us to work together in ways that are mutually satisfying?

Finally, the facilitator would pose "change" questions to the team to set a future direction. He or she might ask you to imagine that you arrive at work tomorrow and discover that a miracle has happened: Compelling communication has become a way of life on our project! Then, you might explore these questions:

- What is different?
- How does it feel?
- What did we do to get here?

(For more examples of positive questions you might ask in a retrospective, see **Positive Questions for an Appreciative Inquiry** in the [Appendix](#).)

Follow a Structure

A retrospective should follow this overall structure³:

1. Get ready.
2. Explore the past.
3. Understand the present.
4. Decide the future.
5. Retrospect the retrospective.

During the *readying*, the facilitator reviews the agenda and the group agrees on ground rules, or guidelines for participation. When you anticipate that emotions might be high, the facilitator must be sensitive to that and spend additional time establishing a "safe" environment, using activities that test for safety. At this time, groups may also review their guidelines for participation. One of the most critical is what Kerth calls the "Prime Directive":

Regardless of what we discover, we must understand and truly believe that everyone did the best job he or she could, given what was known at the time, his or her skills and abilities, the resources available, and the situation at hand.⁴

The readying step is also the time for team members to make a personal commitment to being open to the outcomes of the retrospective. (For questions a facilitator and team would explore to establish retrospective

ground rules, see **Defining Ground Rules for a Retrospective** in the [Appendix](#).)

In the *past* step, the group examines data from the iteration or project they just completed, reviews significant events, appreciates what worked, and analyzes why things happened. During this phase, the facilitator will ask such questions as, What happened? How did you react? Why did it happen? For example, team members might look at the data around their architectural prototype and the portion of the requirements they have selected to prototype. They may recall one or more prototype review sessions with the customer and bring in e-mails that have provided feedback from customers off site. They examine their personal feelings and what happened in the team, how communications transpired, and so on.

During the *present* step, the group interprets what happened by asking, "What did we learn?" and "What do we do well?" They may play out scenarios to discover other ways they might have tackled the iteration, speculating on whether and how the outcomes might have been different. For example, one team speculated about how their iteration outcomes might have improved if they had obtained scenarios for their use cases from a broader customer group, rather than relying on a single customer representative. The representative readily agreed, and they discussed why she used a solo strategy. The team adjusted its approach for the next iteration, to promote greater customer buy-in and save development time. Another project team speculated on how reorganizing the team roles and involving testing earlier might have prevented some of their beta test problems.

The present step of the retrospective prepares the team to transition to *future* thinking and to explore the question, "Now what?" The group examines its strengths to leverage them and plans how to overcome its weaknesses for the next iteration. For example, one early iteration retrospective resulted in decisions to add more glossary terms to help customers delineate business rules, add more rigor to the nonfunctional requirements, have customers use a more rigorous decision rule process for prioritizing their use cases, and insist that management not add new people to the team in the next iteration.

The retrospective ends with, well, a retrospective of the retrospective itself. The team takes five or ten minutes to reflect on the value, productivity, and quality of their interaction in the retrospective session itself, including giving feedback to the facilitator. Incorporating a ritual of this nature into every group gathering is essential to healthy teamwork.

(For a summary list of questions to ask and answer, see **Questions for Each Step of a Retrospective** in the [Appendix](#).)

Base Changes on What You Learn

Retrospectives provide a structure for developing teamwork, and they're the best way to implement sustainable team change. But unless you put what you learn from them into action, your investment in retrospectives

will not be realized. In other words, don't waste your time with retrospectives unless everyone -- including management -- is committed to taking action.

When they leave, all participants in the retrospective should have a clear understanding of what specific actions or behaviors need to be sustained or changed. That means your retrospective should deliver an action plan in the "decide the future" step. For example, during its iteration assessment/retrospective for its second Construction iteration, one project team created an action plan for correcting critical problems with system tests, reestablished how and when team communications would occur, and redefined several key roles, including that of the project sponsor.

Get management support up front for the team to make such changes. Ask managers to participate in each retrospective, to explicitly empower the team to make changes as it chooses, or by creating a set of "Recommendations to Management" for their immediate review and ratification. On one project, we had senior managers come to the last ten minutes of our iteration assessments to see the outcomes of our retrospective recommendations. This gave us immediate sponsorship for both the retrospective process and the specific actions we needed to take.

Making the Case for Retrospectives

To realistically position retrospectives as the structure for your RUP iteration assessments, it's a good idea to share their benefits and barriers with your team and management. Table 2 can help you get started.

Table 2: Analyzing Retrospectives as a Project Management Tool

Benefits
<ul style="list-style-type: none">• Develops a team and project culture that values open and honest feedback.• Fosters agility: The team learns to take corrective actions sooner and continually, thereby increasing the likelihood of project success.• Exploits what we know about adult learning -- provides immediacy, relevance, and self-direction.• Can clarify goals, roles, and communication needs on the project.• Establishes repeatable, successful behaviors.• Seeds an organization with people who bring healthy learning experiences from earlier projects.• Provides opportunity to change the stories, or myths, of the organization, project, or team culture.• Builds collective ownership of project outcomes and processes.• Allows you to make process improvements by fixing known problems.

Barriers

- People will be negative or seek to blame; some will become emotionally upset.
- People will want to rush through an abbreviated iteration assessment instead of doing a full retrospective if they are in a time crunch. Retrospectives should be built into the work plan.
- Managers may fear that if the retrospective is not done well, it will breed cynicism.
- Managers and team members may have fears about airing dirty laundry, especially if that runs counter to the company culture.
- Some team members may feel retrospectives focus only on what is wrong, without sustaining what is working.
- There's a risk of losing team members who may not be willing to look inward or act as team players.
- Change is difficult to implement, and some team members may resist it.
- Reveals team, individual, and management accountability, which may be threatening if it's not part of the company culture.
- If you don't follow up on process improvement resolutions you formulated during the retrospective, then managers and team leaders will lose credibility with the team.
- Takes effort to quantify the return on investment.

Perhaps the easiest route to convincing management that retrospectives are valuable is to talk about your most recent iteration deliverable. You know the one I mean. It's the one that says *if you had known then what you know now, you would have saved days or weeks*. Calculate the cost of gathering the team for some three hours against the time you could have saved, and the business case becomes obvious.

End Well to Begin Well

Ironically, pausing to do in-depth retrospectives permits a project community to adapt and effectively speed up. A key intent of iteration assessments in RUP is to provide learning and self-correcting feedback. Structuring iteration assessments as retrospectives gives team members a specific way to review, play back, and think reflectively about not only how the technology is working, but also how the group process is working (or not working), and this kind of learning is essential to ongoing success. Retrospectives help the group to become more self-sufficient and productive more quickly. They also promote internal and public commitment to corrective action and build not only better software but also a healthy project community.

Appendix: Conducting a Retrospective

[View the Appendix](#)

References

Andreas Birk, Torgeir Dingsoyr, and Tor Stalhane, "Postmortem: Never Leave a Project without It," *IEEE Software*, vol. 19, no. 3, pp. 43-45, May/June 2002.

Bonnie Collier, Tom DeMarco, and Peter Fearey, "A Defined Process for Project Postmortem Review," *IEEE Software*, 1996.

Norm Kerth, *Project Retrospectives: A Handbook for Team Reviews*, Dorset House, 2001.

<http://www.retrospectives.com>

Notes

¹ See Norm Kerth, *Project Retrospectives: A Handbook for Team Reviews*, Dorset House, 2001.

² Victor R. Basili and David M. Weiss, "A Methodology for Collecting Valid Software Engineering Data". *IEEE Transactions on Software Engineering* SE-10 (November, 1984):728-738.

³ Adapted from Norm Kerth, *Project Retrospectives: A Handbook for Team Reviews*. Dorset House, 2001.

⁴ Norm Kerth, *Op.Cit.*



For more information on the products or services discussed in this article, please click [here](#) and follow the instructions provided. Thank you!