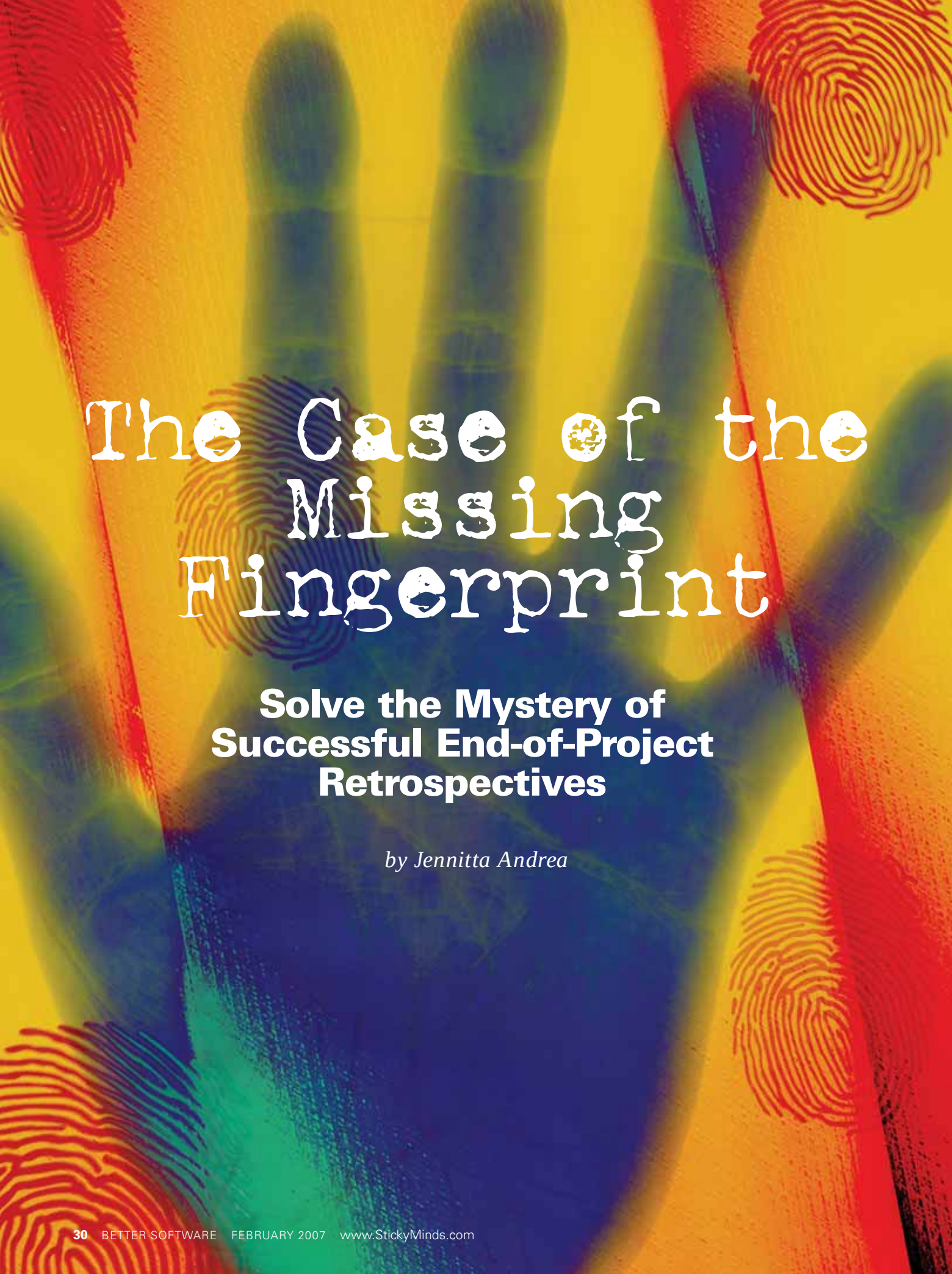


The Case of the Missing Fingerprint

Solve the Mystery of Successful End-of-Project Retrospectives

by Jennitta Andrea

The spreadsheet mystery

How much impact can a single decision have on a project? This is the story of a planning spreadsheet: the impact it had on three different projects, how the teams diagnosed their problems, and what they did about them.

All eyes were on Alpha, the first project in the company to use an agile approach. By all accounts, it was a resounding success—the customers were delighted with the product, schedule and budget commitments were met, and the team was proud of its work. Iteration and milestone retrospectives were conducted faithfully, resulting in incremental improvements to both the work environment and the process over the life of the project. Everyone gave the agile approach "two thumbs up."

The team organized an end-of-project retrospective to capture the important lessons learned during this pilot project. The deliverables of the retrospective were a list of things that worked well (recommended for use in other projects) and a list of things that did not work well (not recommended for other projects). The Alpha graduates then were dispersed within the company to seed three new projects armed with agile experience and the valuable lessons learned.

How did the second-generation agile projects do? Unfortunately, they all had difficulties. Project Beta consistently failed to finish the user stories planned for an iteration, raising concerns about whether the core functionality would be delivered on time. Project Charlie struggled with long, drawn-out iteration planning meetings that often concluded with sketchy plans at best. Project Delta experienced tensions between over and underworked team members due to an unbalanced workload. Rather than adopting symptom-specific remedies (such as lengthening the iteration for Beta, lengthening the planning meeting for Charlie, or changing the team on Delta), the Alpha graduates encouraged each project to conduct an interim retrospective to uncover the root cause of their problems.

In each case, the iteration planning process was found to be the core problem. Beta's completion problems were due to the lack of visible priority of the user stories, resulting in team members' working on too many stories in parallel. Charlie's iteration planning problems were due to the poor status recording in the spreadsheet, forcing status discussions to be held at the start of the planning meetings. Delta's work-balancing problems were due to the rigid assignment of members to tasks during the planning meetings, which became a barrier to members' sharing the workload evenly when estimates didn't match reality. Ironically, the planning and tracking practices were adopted by each of these projects because they were highly rated on Alpha's "what worked well" list. The solution for Beta, Charlie, and Delta was to transition to a more visible, organized, and flexible paper-based planning and tracking process. When they did, immediate improvements were realized.

Why didn't Alpha's planning process work for Beta, Charlie, and Delta? The problems for these three projects actually began during Alpha's end-of-project retrospective. The fundamental mistake was that while the retrospective identified things that worked well and things that didn't, it neglected to tie successful processes to the project context. We know the things that worked on Alpha but not *why* they worked. The mistake was compounded when the second-generation projects didn't consider their unique contexts prior to adopting Alpha's processes.

This article describes an approach to conducting an end-of-project retrospective that captures both outcome and context. I'll apply these forensic techniques to Alpha to show how the project context, like a fingerprint, will help solve the spreadsheet mystery. I'll also advise how new projects can proactively use these retrospective outcomes.

End-of-project retrospective

An Agile Manifesto principle states, "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly." Interim retrospectives (see the StickyNotes for more information) are opportunities for continual learning and process improvement throughout the life of a project. In contrast, an end-of-project retrospective is an opportunity to capture lessons that can be applied to future projects, typically comprising different people, using different tools, and building something completely different.

The following end-of-project retrospective agenda can generally be covered in two to four hours, depending on the team size, project nature and duration, amount of project introspection done to date, etc.

Offer appreciations

Start the retrospective on a positive note with one team member's offering appreciation to another team member for something specific he did on the project. The recipient of the compliment then offers his appreciation to someone else. The process continues until everyone has been thanked and has had a chance to express thanks at least once. The positive energy generated by this activity helps sustain the team throughout the remainder of the session.

Give the project a grade

List the success criteria for the project. Some of the criteria come from the business case or project charter and should have been measured along the way (e.g., specific changes to business processes/workflow and specific improvements to company profitability). Others relate to project execution (on time, within budget, satisfied customer, and defect rate). Still others relate to the project experience (professional growth, work-life balance, stress level, teamwork, and workmanship). If team members did not define the success criteria at the outset, they brainstorm it now. The group gives the project a final grade for each item. These grades will be used during the "collect the wisdom" phase of the retrospective.

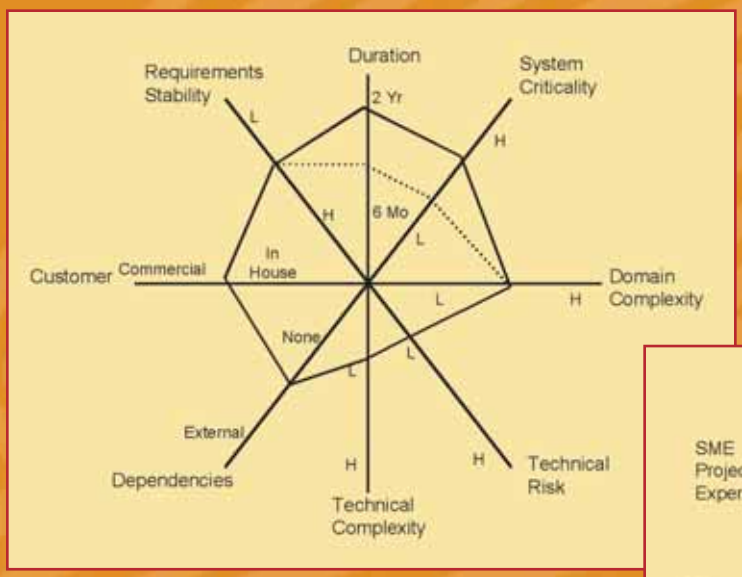


Figure 1: Alpha project fingerprint (radar graph)

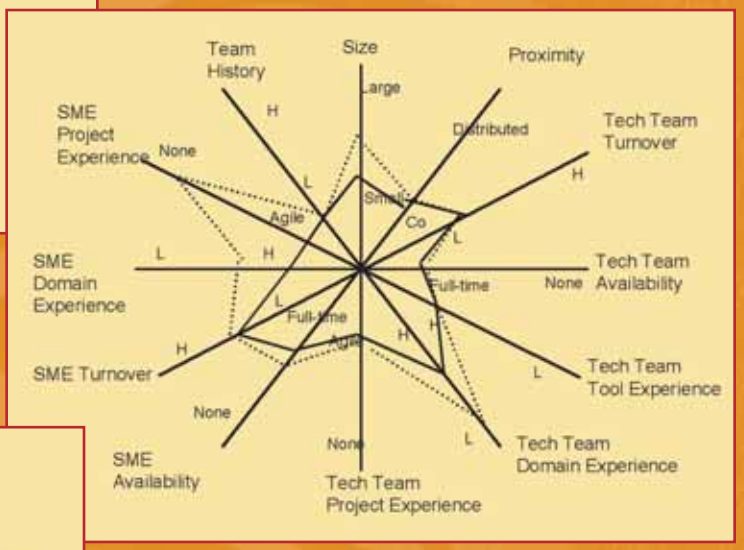


Figure 2: Alpha team fingerprint (radar graph)

Figure 3: Alpha process fingerprint (radar graph)

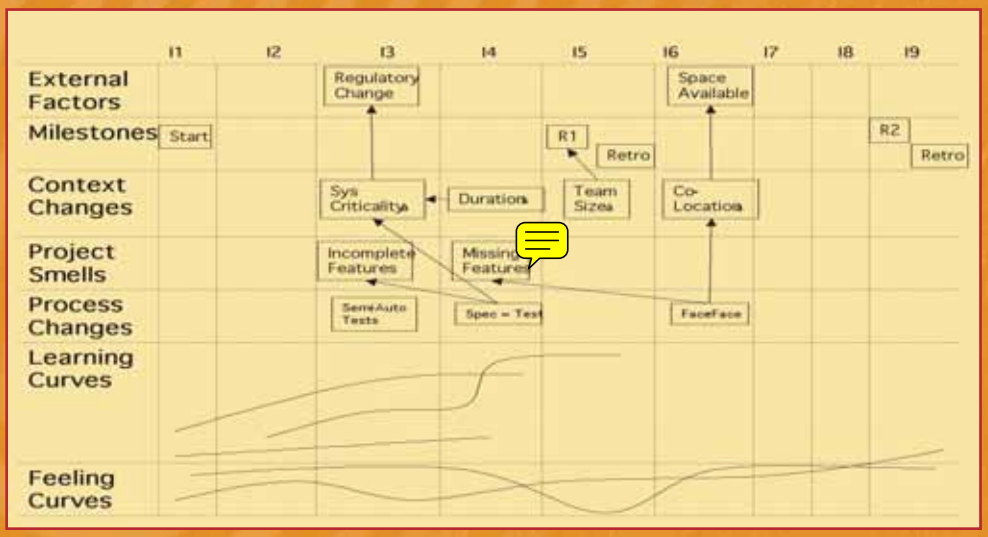


Figure 4: Alpha timeline analysis

Draw the project fingerprint

The project context can be captured visually on one or more radar graphs (see the sidebar Interpret Context-Process Relationships with Radar Graphs). Each axis on the diagram represents a single factor of the project context. Because it is not practical to capture all the project factors on a single radar graph, it is recommended that the factors be grouped together into at least three different graphs that cover project factors, team factors, and process elements (see figures 1, 2, and 3 for the Alpha project). The units along each axis are arranged such that the ideal rankings are located closest to the center of the graph, and the least ideal rankings are along the outermost edge. The graphs in figures 1, 2, and 3 are calibrated for an agile approach.

When multiple points in time are recorded on the same graph, the radar graph begins to resemble the concentric swirls of a fingerprint, so we call them fingerprint graphs. In these examples, the dashed line represents the start of the project and the solid line represents the end of the project. The relationships between the lines enable us to visualize how the project changed over time.

We can see when the process moved toward the ideal rating for a factor because the final rating (solid line) is closer to the center than the initial rating (dashed line). Figure 2 shows that most of the team factors improved, many of them because it is natural for team members to gain experience with the domain, tools, and process during the project. Figure 3 shows that the primary communication channel evolved from email and phone calls to face-to-face discussions, and that the requirements specification technique evolved from examples to more rigorous functional tests.

Furthermore, we can see when the process moved away from the ideal rating for a factor because the final rating is farther from the center than the initial rating. Figure 1 shows a fairly substantial increase in the project duration and system criticality.

Finally, we can see when the process remained stable for a factor because the lines completely overlap. In figure 3, for example, the use of a tool for project tracking was constant throughout the project.

While the fingerprint graphs are an effective way for the team to visualize what changed on the project, they say nothing about when the change occurred, why the change was made, or how the project was impacted. The graphs are the basis for the next steps that dig deeper for these insights.

Perform fingerprint analysis

Recording significant project events along a timeline is an effective way for a team to view the project from multiple perspectives at once and identify patterns and opportunities for process improvement. One style of project timeline for an end-of-project retrospective is described here. The goal of this timeline is to facilitate discovering significant relationships between context changes and process changes.

Create a large grid with the columns labeled with time blocks (i.e., iterations), and the rows labeled as follows: external factors, milestones, context changes, project smells, process changes, learning curves, and feeling curves (see figure 4 for Alpha's timeline). The timeline is populated with content (note cards) and relationships (connecting lines between cards) during the retrospective as follows:

External Factors: Even when risks are identified early in the project and mitigation strategies are developed, external factors can significantly affect the outcome. Unexpected disruptions may come in the form of regulatory changes, corporate acquisitions, or acts of nature. List all anticipated and unexpected external factors on the timeline to mark the point at which they impacted the project.

Milestones: Put a note card on the timeline for each key project date.

Context changes: Review the project and team fingerprint graphs, placing a note card on the timeline for each context change. Briefly perform root cause analysis by repeatedly asking why. For example: The project's duration increased. Why? Because scope and testing time were added to the project. Why? Because system criticality was elevated partway through. Why? Because a new regulatory requirement significantly impacted the project. (See the StickyNotes for details on the "Five Whys" root cause analysis.) Draw lines on the timeline to connect the change to the triggering events. If the context change is triggered by an outside factor, place the note card in the external factors lane.

Project smells: The smell of smoke alerts us to the danger of fire; we must respond to the danger (get out of the building), not to the smell (spray air freshener). Similarly, project smells are symptoms that alert us to a broken or ill-fitting process. Add project smells that were not identified during the process change root cause analysis. An unlinked smell may be the result of an incorrect initial context or a context change that wasn't addressed by a process change.

Process changes: Perform root cause analysis with the process fingerprint graph, placing the note cards on the process changes lane. If the process change was triggered by a project smell, place the note card in the project smells lane (e.g., the specifications evolved to more rigorous functional tests, because some of the features were not fully implemented, because the original examples glossed over key details).

Learning curve: Typically, software development involves an element of learning within the business domain, the toolset, and the development process. Team members draw their domain, tool, and process learning curve on the timeline.

Interpret Context-Process Relationships with Radar Graphs

A process is a collection of decisions about what to do, how to do it, and when to do it. The best decisions are not random but are driven by a clear set of values and by the context to which the process will be applied. When our context is complex, it is helpful to organize our thoughts visually with two or more radar graphs—context on one graph and decisions on another graph.

To understand how to use this visual tool, let's consider a simple process like packing lightly for a trip. Figure A shows the context radar graph, which has a separate axis for each factor. Each axis is labeled independently and has the optimal solution for our goal closest to the middle. For example:

- *Hot climate* is closest to the middle because we'll pack lighter things than for a cold climate.
- *Bike transportation* is closest to the middle because we must pack fewer things than for a car trip.

Figure B shows the decision radar graph. Again, each factor has a separate axis and is labeled with the optimal solution closest to the middle.

There are interrelationships between context factors and decision factors. For example, figure A and figure B show how changing the accommodation constrains the decision about what to pack; we must pack more things as accommodations become more rustic. For constrained relationships, the context and the process radar graphs should have similar footprints (e.g., as the accommodation settings move outward, we expect the articles settings to also move outward. If they don't, there's a chance that you failed to pack something significant).

Figures C and D show how changing the mode of transportation enables the decision about how much to pack. For enabling relationships, the graphs do not have to have similar footprints; just because you can pack more doesn't mean you will.

Values impact decisions and act as a wild card. Two people in the same context may make different decisions because they have different values. For example, someone who values a good night's sleep may bring his own pillow to a hotel, even though the context doesn't require it.

The effect of project and team context on a software development process is far more complex and is subject to many more wild cards than packing for a trip. The value system is rooted in the philosophy of the process. Radar graphs vary from project to project, as they are calibrated for the value system embodied in the target process. Inter-relationships between context and process are very complex and often contradictory.

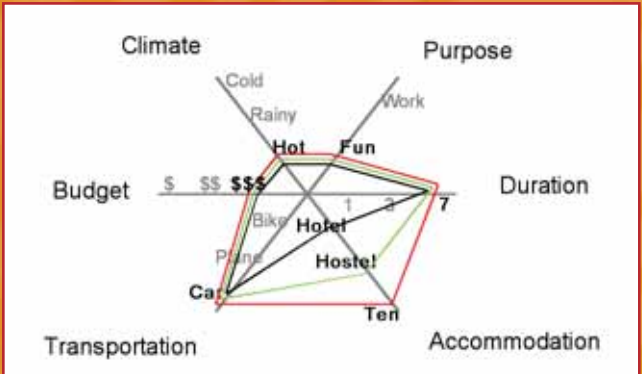


Figure A: Context varying accommodation

Figure B: Decisions affected by accommodation

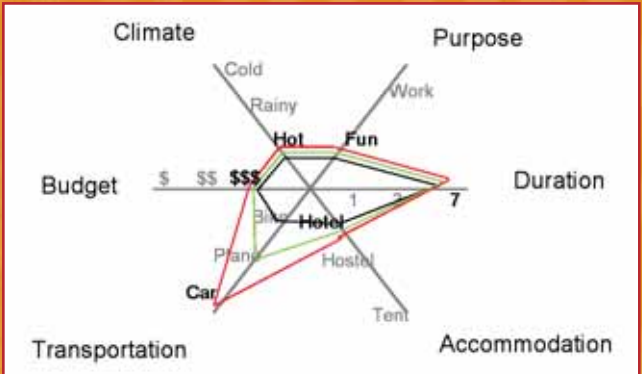


Figure C: Context varying transportation

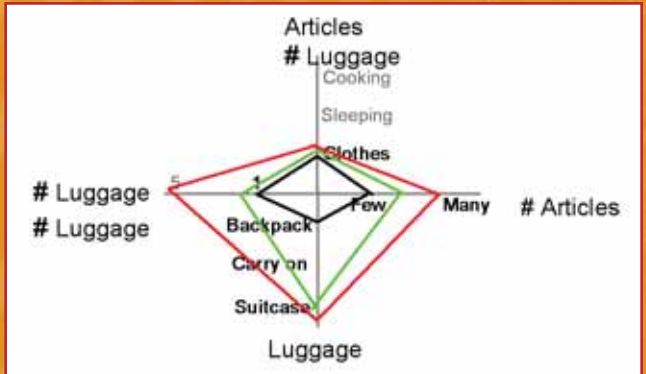


Figure D: Decisions affected by transportation

The curve starts at their initial level (novice, intermediate, and expert) and moves higher as learning takes place.

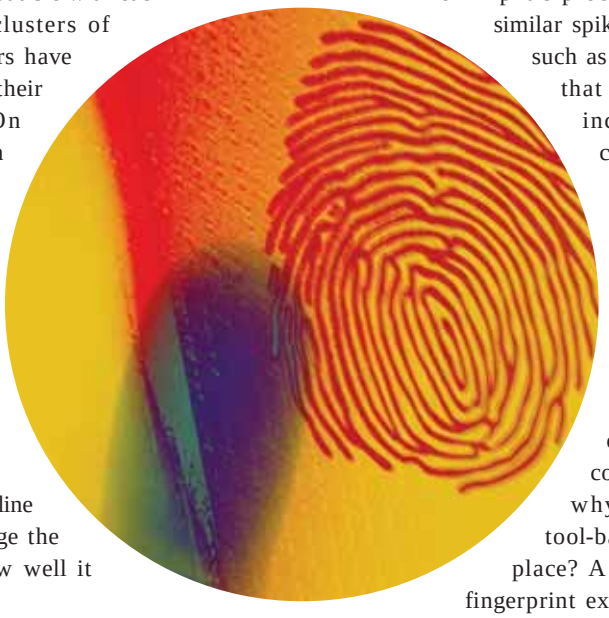
Feeling curve: People's feelings about the project vary over time as the project unfolds. Team members draw a curve that tracks their emotional experience during the project, ranging from high (things are going well for me) to low (things are not going well for me) in relation to events on the timeline.

Collect the wisdom

The final step is to mine the retrospective artifacts—the report card, the fingerprint graphs, and the timeline—for enduring lessons about what worked well, what didn't work well, and why. The team should step back from the details in order to discover general patterns that will be of interest to people outside of the project.

Pay close attention to how the events map to the feelings curves. The team's emotional state is closely linked to the project outcome: A happy team is a productive team; when morale suffers, the project suffers. It's important to understand the causal relationships between events and feeling curves. Explore the points in time where feeling curves diverge (highs and lows at the same point in time) to understand if the process served all team members well.

Look at the end-of-project lines on the fingerprint graphs to see if the overall shapes are compatible with each other. See in the sidebar that clusters of context factors and process factors have similar radar graph footprints if their relationships are constrained. On Alpha's process fingerprint graph (see figure 3), the eye is drawn to the large outward spike along the tracking axis. This spike is not consistent with factors that influence the tracking decision; for example, rankings for team size, proximity, iteration length, planning, and communication hover close to the center of the graph.



Take a high-level look at the timeline to get a sense of how much change the project experienced. Explore how well it responded to change:

- Was the project impacted by external factors? How well did the project respond to the impacts? Could the team have been more responsive?
- How many factors experienced project smells? Were interim retrospectives and feedback cycles effective?
- How much did the context change? Was the change clustered, or was it constant? Were process changes reactive or proactive? Are there any unlinked changes (change for the sake of change)?
- How significant were the learning curves? Was the level of expertise appropriate for the domain/technical complexity, project duration, and turnover? Did the learning curves impact progress?
- What was the impact of all of this change on team morale? How did team members respond to what was going on in the project? How did they respond to other members' feelings?

The "what worked well" list should capture the relevant details about factors that did not need to change, factors that changed in response to context changes, and items that scored high on the project report card.

The "what didn't work well" list should capture the relevant details about mismatches between the factors; factors that changed because of project smells; unlinked project smells, context changes, and process changes; and items that scored low on the project report card.

The plot thickens

By now the astute sleuth has solved the spreadsheet mystery and can clearly articulate why the iteration planning approach and tracking spreadsheet used on project Alpha did not work on Beta, Charlie, and Delta. If you haven't cracked the case yet, go back and study the diagrams one more time before reading on.

The primary clue is the large spike along the tracking axis on Alpha's process fingerprint graph. The lack of similar spikes associated with related factors, such as team size and proximity, indicates that the tracking spreadsheet may be inconsistent with Alpha's project context. But didn't the Alpha team list the spreadsheet as something that worked well?

When the plot thickens, a seasoned detective looks for what is *not* there. Specifically, given the apparent context mismatch, why aren't there process smells associated with iteration tracking on Alpha's timeline? With its ideal context and agile project experience, why did the Alpha team choose a tool-based tracking approach in the first place? A second careful look at the team fingerprint exposes the hidden clue—the initial inward spike associated with team history. Typically, software projects build the team as they build the software; projects typically start on the "low" end of the team history axis. Unlike most software teams, Alpha's technical members had significant prior experience working together; they started on the "high" end of the team history axis. They developed the tracking spreadsheet themselves, had used it before, and were all committed to the discipline needed to make it work.

Although the spreadsheet wasn't an ideal choice for Alpha's overall context, it worked well enough for the team because of its history and commitment—Alpha *made* it work.

This is an example of a wild card affecting process decisions. The Beta, Charlie, and Delta teams were more typical of the software industry and had little or no prior history as a team or with the tracking spreadsheet. After reviewing the facts, it's really no surprise that the project smells emerged on Beta, Charlie, and Delta.

Case closed

The end-of-project retrospective is an important ritual for celebrating the completion of a project and reaching closure on the shared experience. It is also a mechanism for incrementally building a corporate or industry repository of process wisdom. The facts, opinions, insights, and analysis captured during the retrospective are turned into wisdom by overlaying context—it is essential to include reasons why. Project advice should be covered with fingerprints.

Like lifecycles in nature, the wisdom collected at the end of one project can nourish a newly forming project. By creating and understanding its own project, team, and process radar graphs at the beginning, a new project can sift through advice and guidelines to find what fits its situation. The team can visually compare its project context to other projects and consider adopting process elements related to areas where their contexts overlap.

Fingerprint graphs and project timelines are tools to help understand a fragment of the complex, interrelated, and unpredictable subject of how people work together to develop software. They help us observe, analyze, and act. Do not try to use them to develop a cookbook approach to process adaptation or to enforce the "one true way" on all projects. Remain open to anomalies and unexpected approaches. Keep sniffing for clues. **{end}**

Jennitta Andrea is a retrospective facilitator, coach, analyst, developer, tester, and instructor with clearStream. Her work on context-driven process adaptation is based on hands-on experience and reflection on more than a dozen different agile projects since 2000. Jennitta is regularly published, speaks at conferences, and is a board member of the Agile Alliance. Jennitta loves murder mysteries, ballet, playing games, and spending time with her family: Jim, Ava, Shark Boy, and Lava Girl.

Sticky Notes

For more on the following topic, go to www.StickyMinds.com/bettersoftware.

- Interim retrospectives
- The "Five Whys" root cause analysis
- References