By Cynthia F. Cohen, Stanley J. Birkin, Monica J. Garfield, and Harold W. Webb

Managing Conflict in Software Testing

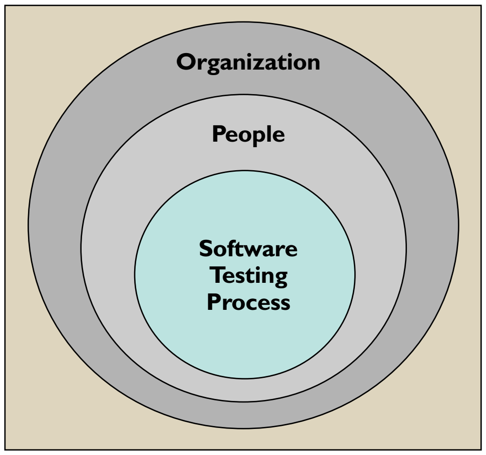
Conflict between software testers and developers is inevitable, but mindful managers minimize its effect on development projects through communication, mutual respect, even social interaction.

Software development involves many people, each focusing on some aspect of the overall process. Software testing is a crucial aspect of ensuring the quality of the end product. Although many tools and automated methods are available, the result ultimately depends on the interpersonal interactions of the people producing the software. However, they often don't all share the same goals or mindset or necessarily feel compatible. Each such factor might thus contribute to conflict, an inevitable part of organizational life managers are constantly trying to resolve [6].

Illustration by Oreste Zevola

Conflicts arise in organizations for a variety of reasons, including scarce resources, interdependent work, differentiated work, competitive reward systems, perceptions of inequity, and asymmetrical distribution of power [4]. While conflict does not always produce negative results and can even enrich organizational outcomes [3], it can certainly disrupt work processes and contribute to poor performance. Software development, like any other work activity, involves many possible sources of conflict that can undermine efficiency. The potential for conflict among software developers and users has been identified [2, 8]. Similarly, conflict can occur within the software development group itself, particularly during the inherently adversarial software testing process.

Because conflict often has the potential to interfere with work performance and product quality, it is important for both scholars and practitioners to identify its sources in software testing, understand how it affects work processes and outcomes, and determine ways to manage it better. In order to understand how best to accomplish these objectives, we conducted in-depth field interviews in 2002 with 10 software testing professionals from four different U.S. companies, two large, one mid-size, one small [1]. These interviews led us to categorize three basic conflict layers—process, people, and organization—and a set of actions available to managers for addressing each of them (see the figure here).



Layers of conflict.

Process (the scarce resource of time). The most frequently mentioned source of conflict by the testers and project managers we surveyed was the allocation of time between development and testing. This is not unique to software testing, as time is a persistent issue in all types of project management scenarios [7]. As organizations of all types strive to quickly get sophisticated, defect-free products to market, time inevitably becomes scarce and more valuable. Testing is often postponed and planned testing time reduced to stay on the delivery schedule.

The result is conflict, as testers and developers compete for time to complete their tasks. The sequential nature of development and testing often results in testers being left with little time. One tester we interviewed said, "They [project managers] let the developers code up to the 11th hour. Then they expected it to be tested. It's very frustrating knowing that once it goes into production there is not enough time to test because they [the developers] were being allowed to develop up to the last minute. I mean literally the last minute—the night before." Another tester said, "I think the developers look at that schedule and see their little piece of time. If it bleeds over, so what? Since they're not at the tail end, they're not the ones burning the CDs and sending them out the door. They don't have the same sense of urgency as the people on the tail end of the project. As development slips, the sense of urgency increases, the stress goes up, and the tension between the two groups starts flying."

Process (user vs. technical requirements). Another source of conflict we identified is the kind of focus developers and testers bring to their jobs. Testers and managers agreed that testers were more focused on user requirements and developers more on the technical nuances of software design. These equally valid but distinctly different views of the process can add value but also create problems. Developers flexing their intellectual muscles often look for novel ways to achieve results. This can lead to application goals incompatible with those of the testers. One example mentioned by a tester we interviewed was "Some of them [developers] embellish on the requirements. Like this one instance where a developer had buttons in his application. There was nothing wrong with the buttons; there was nothing wrong with the way they worked; nothing wrong with the way they looked. But he takes it upon himself, goes out and finds some new buttons, implements them, and they don't end up working on the browser I'm using. Then he changes them back to the way they were because he realizes a lot of users are going to have my version of the browser. Doing that, he forgets about some of them. So there's a couple that don't work."

People (different strokes). Not only do testers and developers perceive the process of software development differently, they often have their own unique mental processes and personality attributes. A tester told us, "Developers think and write the app the way it should work. And they can't understand why anybody would use it other than the way it should work. It's an entirely different mindset as to what QA [quality assurance specialists] feels is an effective unit test and what development feels is an effective unit test." A manager told us, "Our testers are geared toward testing; they have an understanding of the business, not just from the software development standpoint but from what occurs out there in real life and what needs to happen from a functional perspective." Many testers described themselves as "compulsive" and "very detailed" and the testing group as "cohesive." Developers were typically described by testers and managers as "very creative," "temperamental," and "individualistic." While these characteristics often mean individuals are well suited to their work, they can also lead to incompatibility when they work together.

People (personalization of code). Many developers view their code as an extension of themselves and thus take it personally when someone finds fault with it. Conflict ensues as errors are detected and communicated to developers.

Most testers cited developer reluctance to accept the existence of errors, saying, "Development wants you to show them six different ways it should or should not run a certain way" and "That's not my code; clearly the test was broken." One said, "There's the usual resentment and the ego of it, that we're beating up their design or their code or whatever." A more extreme reaction may result in personal attacks and the lodging of complaints against co-workers. One tester said, "I got dinged by one of the developers who thought, for whatever reason, I had something against her. She filed complaints with the manager about me. She said I wasn't doing my job, that I was hindering the whole development process. But the problem was entirely on her end."

Organization (power and politics). Though most organizations recognize the need for high-quality testers and their specialized skill set, testers still struggle to win the respect they deserve. One manager told us, "If you had a diagram with God at the top, the engineers [developers] would put themselves above that." Many testers feel they struggle to maintain their place relative to that of developers. Another tester said, "We have to fight to keep ourselves equal; we are continually doing things to make ourselves feel just as worthy." The lack of status and support makes the tester's job more difficult and time consuming, as the struggle for recognition becomes part of the job itself.

	Sources of Conflict	Managing Conflict
Processes	Scarce resource of time	Time management Plan for schedule overruns Manage effect of schedule changes Learn from project experience
	User vs. technical requirements	Common goals Align individual goals with process metrics Value team more than individual success
People	Different strokes	Team building Train in conflict resolution Sponsor group activities Support informal social contact
	Personalization of code	Understanding one another's point of view Design jobs to support mutual understanding Involve testers in requirements planning
Organization	Power and politics	Structure for success Co-locate teams Integrate development/testing functions Instill ownership
	Managers matter	Involved leadership Create collaborate atmosphere Model effective conflict management

Managing conflict at the source.

Organization (managers matter). Similarly, the manager's role in setting the tone for the importance of testing was also discussed by our interview subjects. One tester said, "I think it starts from the top; they have to support QA. Developers see that QA is not getting the support and the leverage in the company, so they won't, in turn, respect what you're doing." Another tester said, "We have a QA manager whose style is kind of adversarial. The first QA manager we got, she did a very good job of building a working relationship [with] the developers. They brought in this new QA manager, and she basically severed all that relationship building." Managers are often brought into conflicts between testers and developers. This can be a treacherous process, leading to results like the one reported by a tester who recalled a situation where "the developers fired the manager."

The Conflict Dynamic

Such candid comments yielded insight into the common sources of conflict experienced by software testers. Conflict also adversely affects work processes, the quality of work, and the quality of work life. Consequently, managers need to address the conflict dynamic. Several ways in which conflict might be managed better are suggested in our interview results (see the table here).

Process (time management). Time itself is a requirement for effective testing. Although a lack of time might be an expectation for many testers, it still causes considerable stress and conflict. One tester said, "We're told to hurry up and wait, wait, wait, wait. And then you're coming in here on Saturdays to test because they [the developers] didn't deliver until Friday. You're being squeezed to the very end." The loss of testing time was interpreted to mean testing was not highly valued. Managers need to schedule enough time for the testing process and refrain from cannibalizing that time when developers miss their deadlines or when user requirements change. In planning software projects, managers can take advantage of lessons learned from past project management experience. One technique that might help with the identification and resolution of time conflicts is called critical chain scheduling, which seeks to avoid the effects of Parkinson's Law, whereby work expands to fill however much time is available. Instead, it emphasizes the completion time of an entire project, rather than the individual activities along the project's critical path. When time overruns occur, time must be taken from the project buffer, a common pool of "delay time" used by all project activities. Continual requests for scarce buffer time not only raise awareness of time estimation problems but maintain the integrity of testing time; time overruns are taken from the buffer, not from testing time.

Process (setting common goals). Conflict often results when testers and developers do not share common work goals. Group and individual goals need to be clearly defined, not only to release software in a timely fashion but to ensure the release of high-quality code. Performance measures need to be reflected in each aspect. As an example of setting common goals, one manager told us that testers and developers were "driven by the excitement of seeing the hottest network in the world with their code running on it; they really want that. QA also wants to see it happen. So their mission is shared, which I think really works." He added that when testers found problems the developers would say, "I'll fix it right away" or " I never thought of that. We need to make sure the test covers that next time." As this case illustrates, the incidence of conflict is reduced when finding defects becomes a collaborative process with the common goal of improving the software.

Another way to align work performance for the sake of meeting organizational goals is by selecting better quantification measures of software testing processes. Using such methodologies as the Software Quality Metrics Methodology (IEEE Standard 1061-1998) [5], managers tailor a set of measures to evaluate development and testing activities aligned with software system requirements. Properly selected measures should motivate managers and employees alike to achieve system requirements rather than focus narrowly on the products of their individual units.

People (team building). Some testers seem to learn (by trial and error) how to approach developers when experiencing conflict. It can be assumed that not all testers are equally adept at this. Although none of the testers or managers we interviewed indicated that conflict-resolution training was offered in their organizations, formal training would clearly benefit work skills for both groups. Better conflict-handling skills not only help resolve today's problems more effectively, they build better working relationships for the future. Conflict-resolution skill building should include communication skills, cooperative problem-solving techniques, and informal mediation techniques.

Another way to manage differences is to foster relationships between testers and developers. Testers and developers who communicate only when problems occur lack a robust social fabric with which to smooth the process. Several testers and managers we interviewed indicated that social contact paved the way to better working relationships with developers. One manager said his company sponsored group social activities outside of work specifically to improve relationships. Participants went to amusement parks and museums and joined in other recreational activities. Benefits accrue from individuals becoming more comfortable with one another and being able to use these back channels to manage problems productively, including informal gatherings after work and in common areas during work hours.

THE LACK OF STATUS AND SUPPORT MAKES THE TESTER'S JOB MORE DIFFICULT AND TIME CONSUMING, AS THE STRUGGLE FOR RECOGNITION BECOMES PART OF THE JOB ITSELF.

People (understanding one another's point of view). Conflict also stems from the fact that developers and testers do not view the software process the same way. More important, they don't always realize that their perceptions don't match. Structured work activities designed to acquaint developers and testers with the nature of each other's work could help prevent some of these problems. Job rotation is one tool successfully applied to improving employees' understanding of the various jobs within an organization. Although strict job rotation might not be appropriate among software testing and development positions, some aspect of job design could still be used to gain similar benefits. Likewise, early and consistent involvement of testing in the requirements planning phase may provide opportunities for joint understanding at a point in the process when both groups have the time to listen to one another and when there is less existing conflict to inhibit communication.

Organization (structure for success). Our interviews identified several structural concepts for addressing conflict. The most popular among testers was physical co-location. When testers and developers worked in separate locations, communication, as well as personal relationships, was impaired, unlike when both groups worked in close proximity. One tester suggested the two groups should be made to work side by side, adding, "I don't care if you're an organization that has great email, network access, and everything else. If you don't have that personal interaction where you can talk to somebody to resolve an issue, that 10-minute issue turns into a full day."

The second structure is fully integrated teams of testers and developers, reducing the intellectual and physical distance between the groups and facilitating joint goal-setting. However, testers working in integrated teams cited many of the same conflict sources as testers in organizations where the functions were separated. This paradoxical outcome highlights the fact that conflict in software projects must be addressed at multiple levels. The testers did feel, however, they had closer relationships with and better access to developers than testers in nonintegrated structures.

Another structural form shown to improve conflict management is the use of integrated teams vested with comprehensive product ownership. In it, team members follow their applications into the maintenance and enhancement portions of the product life cycle. As the manager of one such organization told us, the team receives the fruit of its labors. This sense of ownership helps establish a common focus for all team members because they know they are responsible for how the application works for end users.

Organization (involved leadership). Finally, acknowledging that developers and testers alike take their cues from the behavior of their managers, managers must signal that testing is an important component of the software development process, not an unrelated activity. Managers must carefully implement performance metrics and goals and provide feedback to development and testing that is congruent with mutually compatible project goals.

Managers also need to understand that conflict between testers and developers is a normal part of the work process. The most notable characteristic distinguishing effective from ineffective management is not the presence or absence of conflict but how effectively it is managed. Managers are role models for conflict-handling styles. Proper performance of this responsibility goes a long way toward solving problems; performing it poorly contributes to the escalation of conflict.

Conclusion

All three layers of conflict—process, people, and organization—are endemic to software development projects. The adversarial nature of software testing and the intellectual and personal differences among testers and developers can be a recipe for conflict. Although it is unreasonable to expect that conflict can be avoided, the goal should be better ways to manage it. As we have suggested, involved managers provide top-level support and set the tone for problem solving. They use organizational structure, management practices, and team-building activities to influence testing activities in ways that improve the overall software development process. **C**

References

1. Barki, H. and Hartwick, J. Interpersonal conflict and its management in information system development. *MIS Quarterly 25,* 2 (June 2001), 195–228.
2. Birkin, S., Cohen, C., Garfield, M., and Webb, H. Causes and consequences of conflict in software testing. Presented at the Global Business and Technology Association 2002 International Conference (Rome, Italy, June 25–29, 2002).
3. Capozzoli, T. Conflict resolution. *Supervision 60,* 11 (Nov. 1999), 14–16.
4. Jameson, J. Toward a comprehensive model for the assessment and management of intraorganizational conflict: Developing the framework. *Int. J. Conflict Mgmt. 10,* 3 (July 1999), 268–204.
5. Modell, C., Ed. *IEEE Standard 1061-1998, IEEE Standard for a Software Quality Metrics Methodology.* Software Engineering Standards Committee, New York, Dec. 1998; see ieeexplore.ieee.org/iel4/6061/16194/00749159.
6. Tjosvold, D. *The Conflict Positive Organization*. Addison-Wesley Publishing Co., Reading, MA, 1992.
7. Umble, M. and Umble, E. Manage your projects for success: An application of the theory of constraints. *Prod. Invent. Mgmt. J. 41,* 2 (2nd quarter 2000), 27–32.
8. Yeh, Q. and Tsai, C. Two conflict potentials during IS development. *Inform. Mgmt. 39,* 2 (fall 2001), 135–149.

Cynthia F. Cohen (ccohen@coba.usf.udu) is a professor in the Department of Management and Organization at the University of South Florida, Tampa.

Stanley J. Birkin (sbirkin@coba.usf.edu) is a professor in the Department of Information Systems and Decision Sciences at the University of South Florida, Tampa.

Monica J. Garfield (mgarfield@bentley.edu) is an assistant professor in the Computer Information Systems Department at Bentley College, Waltham, MA.

Harold W. Webb (hwebb@ut.edu) is an assistant professor in the Information Technology Management Department at the University of Tampa.

This work was supported in part by The National Institute for Systems Test and Productivity at The University of South Florida in Tampa under the USA Space and Naval Warfare Systems Command, Grant No. N00039-01-1-2248.