developer.*
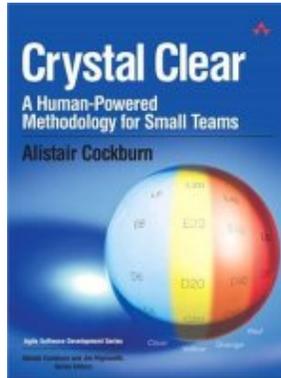### The Independent Magazine for Software Professionals

Book Review

Crystal Clear:
A Human-Powered Methodology for Small Teams

Addison-Wesley, 2005, 336 pages (ISBN 0-201-69947-8)

By Alistair Cockburn

Review By Donna L. Davis

With the deluge of software development methodologies and mindsets touted in recent years, the choice hardly seems crystal clear. In fact, Alistair Cockburn's book, Crystal Clear: A Human-Powered Methodology for Small Teams may seem to muddy the water further. That is, until you read it, and discover Cockburn's common-sense approach, borrowing the best-in-class from different camps.

What's In a Name?

Crystal Clear is just one among many in the Crystal "family" of methodologies, which Cockburn describes as having a "common genetic code." The methodologies are distinguished by size and criticality, analogous to the mineral properties of color and hardness. Hence, Crystal borrows its name from quartz, ranging from Clear, for "collocated teams of 8 or fewer people," Yellow, for teams of 10-20, Orange, for 20-50, Red, for 50-100, and Maroon, Blue, and Violet covering greater numbers. Is the quartz/methodology analogy clever and useful, or bewildering and contrived? That probably depends on who you ask and their propensity to cynicism.

Essential Properties

Cockburn's tolerance for extensibility and flexibility offers practitioners a foot-hold for transitioning toward agile practices. He acknowledges that in the trenches of development, some of the more extreme agile initiatives cannot be rigidly adopted in every environment. Through interviews and case studies, he highlights common denominators of successful project teams, which he breaks down into seven essential properties. (Note that the adjective "essential" is not happenstance. It replaces the formerly popular term, "best," with aspirations of sufficiency, not grandeur.)

1. Frequent Delivery

Cockburn describes frequent delivery as, "the single most important property of any project, large or small, agile or not." It's hard to argue with the advantages of frequent delivery:

- it provides an opportunity for feedback from clients;

- it gives sponsors a warm-and-fuzzy affirmation that work is actually progressing;

- it provides the project team an opportunity to debug and work out deployment processes along the way, and perhaps most importantly; and

- it gives developers a sense of accomplishment.

Delivery can (and many times must) take different forms, depending on the nature of the project, client, and environment. When a client simply cannot accept delivery of incremental software updates, Cockburn suggests deploying to a test workstation, ideally with a friendly user trying it out. When even that is not possible, he recommends incorporating user viewings so the users have an ample opportunity to see work-in-progress.

How frequent is frequent? Anywhere from an hour to three months, with the average falling somewhere between two weeks and two months.

2. Reflective Improvement

A buzz-phrase from the quality movement was "continuous improvement." Reflective improvement is along the same lines, but as the name implies, involves the team getting together periodically throughout the project to think about and discuss what is and is not working.

The idea is to not wait until the project is over to conduct a project retrospective, hoping to learn something helpful for the next go-round, but to adjust and regroup as necessary throughout the project in nimble fashion. Cockburn recommends devoting an hour every few weeks or month for this purpose.

3. Osmotic Communication

"Osmotic communication" is not only the coolest new catch-phrase, but a way of referring to the sort of indirect information flow that occurs when you overhear background discussion. (Didn't that used to be called "eavesdropping?") The real point of this property is to emphasize the importance of team members being co-located, either in the same room or in adjacent offices.

Cockburn acknowledges the possible pitfalls—namely the occasional need for private space for personal communication, as well as the need for silence during concentration, and offers some office space configurations that make accommodations for these factors.

4. Personal Safety

The personal safety property refers to trust and fostering good relationships between team members so that genuine communication is not hindered by fears of revealing inadequacy or reprisals for making a mistake. Cockburn emphasizes the importance of not confusing the sort of "amicability" he refers to here with stilted politeness, which can obscure disagreements and cause more damage.

5. Focus

Do you know what your top two priorities are at any given point in time? You should, says Cockburn, if you are focused. Yet, knowing what to do is not the only key. You must also have time and peace of mind to work on the priorities. Therein lies the rub. Many developers are assigned multiples projects simultaneously, and the loss of focus due to transition wreaks havoc on productivity. According to the interviews Cockburn conducted, one and one-half projects is the most a person can handle and remain effective.

6. Easy Access to Expert Users

In an ideal world, an expert user would be co-located and available full-time for direct involvement throughout the design and build. In real life, that's rare, but if an expert user is not available at least one hour a week, the likelihood of a successful project will diminish. Cockburn recommends weekly or semi-weekly user meetings with phone calls as needed. If

a user isn't available to be directly involved with the project team regularly, he suggests sending a developer to shadow a user and become a trainee for period of time.

7. Technical Environment

Successful project teams operate in a technical environment with a core set of tools and practices: automated tests, configuration management, and frequent integration. While Cockburn is reluctant to say that automated tests are absolutely necessary for Crystal Clear (since adequate manual tests can be used) he makes the astonishing claim that every programmer he has interviewed who has moved to automated tests declared he would never work without them again.

A configuration management system for controlling versions of code should be considered "the most critical noncompiler tool," according to Cockburn's research. Since integration of disparately developed code components is traditionally a trouble area, Cockburn cites the importance of integrating frequently, while the code is still fresh in the minds of developers, and to prevent too many errors from accumulating.

Summary

Crystal Clear, the methodology (or "methodology generator", as Cockburn prefers to characterize it) is simple and makes sense. Crystal Clear, the book, is written accessibly, first summarizing the seven properties of successful projects, then detailing specific useful techniques gleaned from actual projects, followed by an extensive question/answer chapter. Alistair Cockburn anticipates and responds to the inevitable critics, providing the sort of feedback one might anticipate from a panel discussion.

Whether one embraces Crystal as the Next Big Thing, or discounts it as a re-branding of common knowledge, Alistair Cockburn deserves credit for his thoughtful, practical dissertation, which brings transparency and clarity to a subject that is becoming increasingly complex.

<p align="center">###</p>

Donna Davis is a database administrator, project manager, programmer, supervisor, and professional author living and working in North Carolina, US. She can be reached through the editorial staff of **developer.***.