

## Novelty in Sameness

**Hakan Erdogmus**

**P**eople frequently point out that they've used or even embraced a hyped approach years before it was "reinvented." But when asked whether they're still employing it, they respond with a dismissive "no" that contradicts their initial proud and eager association. Sometimes in the same breath, they would trash the approach with an arsenal of harsh criticism. Such paradoxical reactions don't surprise me any more, for new ideas in our industry invariably carry a flavor of sameness, a feel of bitter familiarity. Especially in a fast-changing industry that is hype-prone, what can't be more natural than a little bit of defensiveness and skepticism stemming from past frustrations with



promised silver bullets, the unconscious fear of lagging behind, and the need to simultaneously preserve previous efforts and belong with the "latest, greatest." And I'm not immune either. I've succumbed to similar reactions upon encountering an alleged new method. On the surface, the method would look strikingly familiar, reminding me of not particularly fruitful things I did years ago, and I'd conclude it's the "same old" dressed in a new outfit, unworthy of further study. But if I persevered, a closer investigation would usually reveal some novel aspect, a difference in philosophy, a new principle not initially apparent.

On one hand, the more things change, the more they stay the same at a certain level. On the other, hyped paradigms, even when they fail to achieve their original grandiose goals, sometimes inconspicuously leave behind an essence that in time morphs and penetrates the mainstream with positive and influential, if not revolutionary, re-

sults. In complex environments, the appearance of sameness can be as deceptive as the appearance of novelty. The latter might be subtle and not instantly recognizable. Also, the context in which ideas come into being and evolve, the intention behind them, and the way they're applied might make a difference. Timing, intention, context, and convergence with trends outside the field can infuse a seemingly old idea with life and inertia, making it worthy of consideration in a new light.

A case in point is iterative and incremental development. I've heard many declare, "Oh yes, sure we do IID." Or they say that what they used to do umpteen years ago is indistinguishable from IID, adding there's "nothing novel about it!" Once in a while these claims are justified, but more often, further inquiry dispels them.

### **What's iterative and incremental?**

Evidently, my understanding of IID isn't the norm, so before I proceed, here it is. I submit that this interpretation typifies the contemporary context in which it has been proposed.

The iterative part implies repeating essentially the same process in several (more than a few) cycles. The duration of these cycles, or iterations, is often the same but can vary. Duration is measured in days, weeks, or months, but not more than a few months and definitely not years. Both phased and iterative development are special cases of staged development, but unlike phases, iterations apply similar principles, employ similar activities in a similar order (if there is one) and manner, and produce similar artifacts or output.

The incremental part implies delivering new functionality in chunks or extending existing functionality in a piecemeal manner. By functionality, I mean working, ideally end-to-end features—not requirements documents, not

### STAFF

Senior Lead Editor  
**Dale C. Strok**  
 dstrok@computer.org

Group Managing Editor  
**Crystal Shif**

Senior Editors  
**Shani Murray, Dennis Taylor, Linda World**

Assistant Editor  
**Brooke Miner**

Editorial Assistant  
**Molly Mraz**

Publication Coordinator  
**Hilda Carman**  
 software@computer.org

Production Editor  
**Jennie Zhu**

Technical Illustrator  
**Alex Torres**

Publisher  
**Angela Burgess**  
 aburgess@computer.org

Associate Publisher  
**Dick Price**  
 dprice@computer.org

Membership/Circulation Marketing Manager  
**Georgann Carter**

Business Development Manager  
**Sandra Brown**

Senior Production Coordinator  
**Marian Anderson**

### CONTRIBUTING EDITORS

**Robert Glass, Warren Keuffel,  
 Keri Schreiner, Joan Taylor**

### CS PUBLICATIONS BOARD

Jon Rokne (chair), Mike Blaha, Angela Burgess,  
 Doris Carver, Mark Christensen, David Ebert,  
 Frank Ferrante, Phil Laplante, Dick Price,  
 Don Shafer, Linda Shafer,  
 Steve Tanimoto, Wenping Wang

### MAGAZINE OPERATIONS COMMITTEE

Robert E. Filman (chair), David Albonesi, Jean Bacon,  
 Arnold (Jay) Bragg, Carl Chang, Kwang-Ting (Tim)  
 Cheng, Norman Chonacky, Fred Douglass,  
 Hakan Erdogmus, David A. Grier, James Hendler,  
 Carl Landwehr, Sethuraman (Panch) Panchanathan,  
 Maureen Stone, Roy Want

**Editorial:** All submissions are subject to editing for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IEEE Software* does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society.

**To Submit:** Access the IEEE Computer Society's Web-based system, Manuscript Central, at <http://cs-ieee.manuscriptcentral.com/index.html>. Be sure to select the right manuscript type when submitting. Articles must be original and not exceed 5,400 words including figures and tables, which count for 200 words each.

models (not even executable models), and not test plans. An increment results in a working subsystem that serves as the starting point for the next increment.

### Modern IID versus related practices

Phased development with feedback loops, prototyping and piloting, and stepwise refinement resemble IID but differ in important ways.

Feedback loops indicate the necessity to rewind a process under unexpected circumstances—for example, in response to a need to modify an earlier model or requirement. The loops are guarded, triggered only when such circumstances arise, and avoided otherwise. Iterations, on the other hand, are unconditional except when the project is abandoned.

Prototyping and iterations are special cases of staged development. Both support learning, risk mitigation, and feedback. However, prototyping is often a special stage repeated a limited number of times, often early in the project or when circumstances warrant. Nor is prototyping necessarily incremental: it often results in a throw-away mock, rather than an actual, evolvable, working subsystem. Similar arguments apply to piloting.

Stepwise refinement is a special case of phased development. It involves successively transforming higher-level models to lower-level models, with the final level representing a deployable system. The refinement process as well as the underlying activities, notations, and artifacts change across abstraction levels. So, stepwise refinement, and model-driven approaches which build upon a similar idea, aren't IID per se.

### Intention is the key

Using IID concepts in a process doesn't necessarily make the process IID, just as any intra-iteration sequencing in an IID process doesn't make it sequential or phased. Granted, a method that's non-IID at the highest level might involve sub-processes that are IID at some lower level. Conversely, IID processes can incorporate phasing, piloting, prototyping, or stepwise refinement within iterations.

Elements of iterativeness and incrementality are indeed found in many

processes, both natural and synthetic, on different levels. But it's important to distinguish between the incidental or necessitated kind from the deliberate and embraced kind. IID in the modern sense is deliberate and embraced; it's a pillar, not an afterthought, not a patch. This distinction, though subtle, constitutes a notable departure from most older applications of IID concepts in software projects. The distinction isn't purely philosophical: it can significantly affect a project's execution and results, for better or worse.

### The gravity-defying waterfall

As an example, consider Winston Royce's account of traditional, phased development, expounded in his 1970 article "Managing the Development of Large Software Systems" (*Proc. IEEE WESCON*, 1970, pp. 328–339). As many point out nowadays, Royce's proposal incorporated concepts with an IID flavor. This statement might seem strange given that Royce's proposal came to be known as the waterfall model, the stereotypical IID archrival. Some even go further, just short of suggesting that the waterfall proponents would have ended up doing IID if only they had understood Royce's nuances, read between the lines, and paid attention to the fine print. (This extreme interpretation seems a bit far-fetched.)

Focusing on similarities between traditional and modern perspectives gives us only part of the picture. A more complete (or different) one might emerge if we focus on the differences. Yes, Royce suggested feedback loops between phases, but he also stated, "*Hopefully*, the iterative interaction between the various phases is confined to successive steps," and further, "*Unfortunately*, for the process illustrated, the design iterations are never confined to successive steps" (italics mine). So it seems he perceived these loops as undesirable, almost a necessary evil with which we must live. Because the loops have implied guards that determine when they're triggered, I'd argue that any appearance of iterativeness is incidental. Royce also suggested repeating the process twice "*if the computer program is being developed for the first time*" (italics mine), initially to prototype the system and then for real. This might sound like IID, but it's many levels removed from

## The *IEEE Software* Boards

Four new members have recently joined the Editorial Board: associate editors in chief Annie Combelles, Sophia Drossopoulou, and Forrest Shull and columnist Jeff Patton. Annie, formerly on the Advisory Board, is president of DNV/Q-Labs and a long-time contributor to *Software*. Armed with over 30 years of experience in software quality and process improvement, Annie will look after the magazine's well-established quality coverage area. Sophia is a faculty member at the Imperial College's Department of Computing in London, with extensive expertise in programming languages. Fittingly, she will build up and oversee a new focus area on programming languages and paradigms. Forrest is a senior scientist and director of measurement and knowledge management at the Fraunhofer Center for Experimental Software Engineering in Maryland. He'll leverage his know-how in assessing software technologies and practices on two fronts: as editor of the upcoming Voice of Evidence department and as associate editor responsible for the *Software's* empirical-results coverage. Jeff is a principal consultant for ThoughtWorks and an acclaimed expert, author, and educator in software usability and interaction design. His new column, User Centric (to debut later this year) will champion approaches that put users in their rightful place: the center of development activity.

Joining the Advisory board this month are Elisa Baniassad, Ward Cunningham, and Markus Völter. Elisa is an assistant professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong. Her research focuses on aspect-oriented techniques and cross-cultural design and programming methodologies, strengthening our human aspects, programming, and design areas. Ward is the Eclipse Foundation's director of Committer Community Development. A pioneer and inventor, his contributions have been numerous and among the most influential in software development (including the patterns movement, Extreme Programming, the wiki, and Fit, to name a few). Markus works as an independent consultant and coach, focusing on software architecture, middleware, and model-driven software development. An author of several patterns and books, he's a sought-after speaker at developer conferences worldwide.


The volunteers who have recently retired from the boards are Don Bagert, associate editor in chief for education and training; Richard Thayer, editor of the Glossary department; and advisory board members Maarten Boasson, Dehua Ju, Tomoo Matsubara, Dorothy McKinney, Susan Mickel, Susanne Robertson, Grant Rule, Girish Seshagiri, and Simon Wright.

I welcome the new members to the *IEEE Software* boards. I sincerely thank the retiring members for their invaluable service and wish them success in their future endeavors. —Hakan Erdogmus

the intentional, embraced repetition that characterizes modern IID. First, two is a very small number as far as iterations go. Second, prototyping, while a great learning and feedback strategy, isn't necessarily incremental, as I previously explained.

No doubt, what Royce proposed was pioneering at the time, and it fit the world of algorithmically intensive scientific applications written in early-generation programming languages. Royce had the foresight to incorporate IID-related ideas (feedback loops and prototyping), but in

contrast with modern IID, his was essentially a sequential, phased model. (Notably, though, his article develops the model in an iterative and incremental way.)

Novelty and sameness can conceal each other. Regardless of how things might appear in hindsight, we risk missing the mark if we indiscriminately insist on seeing new proposals and emerging movements simply as reincarnations of old advice, whether to justify or refute them. Do you disagree? Write to me at [hakan.erdogmus@computer.org](mailto:hakan.erdogmus@computer.org). 

## EDITOR IN CHIEF

**Hakan Erdogmus**

[hakan.erdogmus@computer.org](mailto:hakan.erdogmus@computer.org)

EDITOR IN CHIEF EMERITUS:  
Warren Harrison, Portland State University

## ASSOCIATE EDITORS IN CHIEF

**Design:** Philippe Kruchten, University of British Columbia; [kruchten@ieee.org](mailto:kruchten@ieee.org)

**Empirical Results:** Forrest Shull, Fraunhofer Center for Experimental Software Engineering, Maryland; [fshull@fc-md.umd.edu](mailto:fshull@fc-md.umd.edu)

**Human and Social Aspects:** Helen Sharp, City University, London; [h.c.sharp@open.ac.uk](mailto:h.c.sharp@open.ac.uk)

**Management:** Stan Rifkin, Master Systems; [sr@master-systems.com](mailto:sr@master-systems.com)

**Processes and Practices:** Frank Maurer, University of Calgary; [maurer@cpsc.ucalgary.ca](mailto:maurer@cpsc.ucalgary.ca)

**Programming Languages and Paradigms:** Sophia Drossopoulou, Imperial College London; [s.drossopoulou@imperial.ac.uk](mailto:s.drossopoulou@imperial.ac.uk)

**Quality:** Annie Combelles, DNV/Q-Labs; [annie.combelles@dnv.com](mailto:annie.combelles@dnv.com)

**Requirements:** Roel Wieringa, University of Twente; [roelw@cs.utwente.nl](mailto:roelw@cs.utwente.nl)

## DEPARTMENT EDITORS

**On Architecture:** Grady Booch, IBM; [grady@booch.com](mailto:grady@booch.com)

**Bookshelf:** Warren Keuffel, independent consultant; [wkeuffel@computer.org](mailto:wkeuffel@computer.org)

**Design:** Rebecca J. Wirfs-Brock, Wirfs-Brock Associates; [rebecca@wirfs-brock.com](mailto:rebecca@wirfs-brock.com)

**Loyal Opposition:** Robert Glass, Computing Trends; [rlglass@acm.org](mailto:rlglass@acm.org)

**Not Just Coding:** J.B. Rainsberger, Diaspar Software Services; [me@jbrains.info](mailto:me@jbrains.info)

**Open Source:** Christof Ebert, Vector Consulting; [christof.ebert@vector-consulting.de](mailto:christof.ebert@vector-consulting.de)

**Requirements:** Neil Maiden, City University, London; [n.a.m.maiden@city.ac.uk](mailto:n.a.m.maiden@city.ac.uk)

**Tools of the Trade:** Diomidis Spinellis, Athens Univ. of Economics and Business; [dds@aub.gr](mailto:dds@aub.gr)

**User Centric:** Jeff Patton, ThoughtWorks; [jpatton@acm.org](mailto:jpatton@acm.org)

## ADVISORY BOARD

Stephen Mellor, consultant (chair)  
Jennitta Andrea, ClearStream Consulting  
Elisa Baniassad, Chinese University of Hong Kong  
J. David Blaine, consultant  
Ward Cunningham, Eclipse Foundation  
David Dorenbos, consultant  
Kaoru Hayashi, SRA  
Simon Helsen, SAP  
Juliana Herbert, ESICenter UMISINOS  
Gargi Keeni, Tata Consultancy Services  
Karen Mackey, Cisco Systems  
Steve McConnell, Construx Software  
Bret Michael, Naval Postgraduate School  
Ann Miller, University of Missouri, Rolla  
Deependra Moitra, Infosys Technologies, India  
Frances Paulisch, Siemens  
Wolfgang Strigel, consultant  
Dave Thomas, Bedarra Research Labs  
Rob Thomsett, The Thomsett Company  
Laurence Tratt, King's College London  
Jeffrey Voas, SAIC  
Markus Völter, consultant  
John Vu, The Boeing Company