Journal of Cost Analysis and Parametrics, 7:149–159, 2014

ISSN: 1941-658X print / 2160-4746 online
DOI: 10.1080/1941658X.2014.987005

ICEAA
www.iceaaonline.com

On the Shoulders of Giants: A Tribute to Prof. Barry W. Boehm

JO ANN LANE[1], DANIEL D. GALORATH[2], and RICARDO VALERDI[3]

[1]University of Southern California, Los Angeles, California
[2]Galorath Incorporated, Los Angeles, California
[3]University of Arizona, Tucson, Arizona

Prof. Barry Boehm's life work is full of contributions to the software engineering and systems engineering disciplines. This article presents Prof. Barry Boehm's work in the context of the giants on whose shoulders he stands as well as the people he has mentored to carry on his work. Much of Prof. Boehm's work described in this article focuses on his key contributions to the software and system development industries, as well as from the perspective of the enduring legacy he has established with his industry affiliates and students.

Introduction

In the 17th century, the French painter Nicolas Poussin produced a painting of two characters looking into the horizon searching for the rising sun. One of the characters, Orion, was a blind giant and therefore needed help from the second character, Cedalion, who was a dwarf but could see in the distance (Merton, 1965). This painting became the inspiration for the expression "to be able to stand on the shoulders of giants." In 1676, Sir Isaac Newton similarly remarked in a letter to his rival Robert Hooke in discussing experiments in optics:

> *What Descartes did was a good step. You have added much several ways, and especially in taking the colours of thin plates into philosophical consideration. If I have seen a little further it is by standing on the shoulders of Giants.*

In many ways, Prof. Boehm has been a giant that has helped others see further, thanks to his leadership and mentorship. Though his industry affiliates, he has influenced how organizations develop software-intensive systems, helping them to better plan and execute successful systems. In addition, over thirty doctoral students at the University of Southern California have established their own intellectual pursuits by building on Prof. Boehm's research.

This article provides a review of Prof. Boehm's academic genealogy to show the genesis of his work, and discusses some of his major contributions to systems and software engineering, then elaborates further to show how he continues his work through on-going collaboration with his own academic offspring.

Address correspondence to Ricardo Valerdi, Department of Systems and Industrial Engineering, The University of Arizona, Engineering, Rm. 225, 1127 E. James E. Rogers Way, Tucson, AZ 85721-0020. E-mail: rvalerdi@email.arizona.edu

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/ucap.

Prof. Boehm's Academic Genealogy: Looking Back to 16th-Century Europe

As mentioned above, Prof. Boehm's academic genealogy can be traced back to mathematicians whose inventions have inspired generations. Prof. Boehm's Ph.D. is in mathematics from the University of California Los Angeles in 1964. Using information from the mathematics genealogy project (American Mathematical Society), we were able to determine that his advisor, Elliott Ward Cheney, Jr., obtained his Ph.D. from the University of Kansas in 1957. Continuing this search, Cheney's advisor, Robert Schatten, received his Ph.D. from Columbia University in 1943. We can trace this lineage of mathematicians back to around 1500 when Nicolaus Copernicus received multiple doctorate degrees from universities in Poland and Italy. In between are several famous names such as Leibniz, Poisson, Lagrange, and Bernoulli, who established the field of modern mathematics with their contributions in partial differential equations, number theory, and fluid mechanics. The partial lineage is provided in Table 1.

Many of the influences from the mathematics heritage are evident in Prof. Boehm's own work and will be highlighted in the following section.

Key Technical Accomplishments

Prof. Barry Boehm's life work is full of contributions to the software and systems engineering disciplines. Many of these contributions resulted from observations of failed projects

TABLE 1 Prof. Boehm's academic lineage

Name	Degree Granting Institution (Graduation Year)
Barry Boehm	University of California Los Angeles (1964)
Elliott Ward Cheney, Jr.	University of Kansas (1957)
Robert Schatten	Columbia University (1943)
Francis Joseph Murray	Columbia University (1936)
Bernard Osgood Koopman	Harvard University (1926)
George David Birkhoff	University of Chicago (1907)
Eliakim Hastings Moore	Yale University (1885)
Hubert Anson Newton	Yale University (1850)
Michel Chasles	École Polytechnique (1814)
Simeon Denis Poisson	École Polytechnique (1800)
Joseph Louis Lagrange	Università di Torino (1754)
Pierre-Simon Laplace	Unknown
Leonhard Euler	Universität Basel (1726)
Johann Bernoulli	Universität Basel (1694)
Jacob Bernoulli	Universität Basel (1676)
Nicolas Malebranche	Unknown (1672)
Gottfried Wilhelm Leibniz	Universität Altdorf (1666)
Erhard Weigel	Universität Leipzig (1650)
Philipp Müller	Universität Leipzig (1604)
Christoph Meurer	Universität Leipzig (1582)
Moritz Valentin Steinmetz	Universität Leipzig (1550, 1567)
Georg Joachim von Leuchen Rheticus	Martin-Luther-Universität Halle-Wittenberg 1535
Nicolaus (Mikołaj Kopernik) Copernicus	Uniwersytet Jagielloński/Università di Bologna/ Università degli Studi di Ferrara/Università di Padova (1499–1503)

as well as efforts to make better those engineering practices and tools that showed promise. In addition, Prof. Boehm has often showed us how to use existing tools, theories, and practices in new ways or to futher extend their applications as new challenges or unexplained phemonena are identified. The following describe instances where Prof. Boehm was able to draw upon the intellectual legacy of some of his more prominent academic ancestors:

- George Birkoff's work in ergodic theory that states that the system that evolves for a long time "forgets" its initial state and has been used to explain entropy for dynamical systems. Its influence can be seen in Prof. Boehm's software cost estimation Incremental Development Productivity Decline (IDPD) factor that describes how software development productivity declines as the maintenance of previous software increments detracts from new increment productivity.
- Poisson's statistical analysis techniques for events occurring continuously and independent of one another (such as software defect rates) upon which COQUALMO (COnstructive QUALity MOdel) builds to evaluate the impact of defect removal techniques and the effects of personnel, project, product, and platform characteristics on software quality.
- Pierre-Simon Laplace's work with the Bayesian interpretation of probability, a key step in Prof. Boehm's cost model development methodology used to develop COCOMO (Constructive Cost Model) and associated derivatives.
- Leonhard Euler's contributions to graph theory that Prof. Boehm and others employ today in many software and system modeling tools used to design, analyze, and understand various aspects of software-intensive systems and systems of systems.
- Johann and Jacob Bernoulli's fascinations with curves such as those that Prof. Boehm often uses to show the "not too much, not too little" sweetspots for engineering activities.
- Gottfried Leibniz's inventions in the field of mechanical calculators that led to software-based calculations such as those in the COCOMO models.
- Georg Joachim von Leuchen Rheticus's passion for triangles can be seen in Prof. Boehm's triangles that show the relationships between software cost, schedule, and quality.
- Copernicus postulated that the Earth was not the center of the universe; so has Prof. Boehm postulated that today no system is an island unto itself, and fixed, baselined requirements are not the core of a successful system—rather, there are additional forces (e.g., adaptability, flexibility, interoperability with other systems, securability, world events) at play on the initial set of requirements which determine the long-term success of a system development effort.

The following takes a look at how many of these influences have come together in two of Prof. Boehm's major engineering contributions that continue to evolve today: cost estimation and the spiral development process.

Cost Estimation

Prof. Boehm's work of the past several decades has centered on transitioning software development from a pure art into an engineering discipline. This discussion touches on a few dimensions of that work. Arguably, his work in software cost estimation has had the biggest impact on the software industry. Table 2, adapted from a paper by Boehm and Valerdi (2008), shows the history of estimating models.

It is interesting to note that Prof. Boehm's COCOMO model was not the first software estimating model. However, Prof. Boehm had widespread influence and has done more for

TABLE 2 Predominant cost models: Initial releases (Boehm & Valerdi, 2008)

Cost Model	Year
SDC (Systems Development Corp.)	1965
TRW Wolverton	1974
Putnam SLIM (Software Lifecycle Management)	1976
Doty	1977
RCA Price S	1977
Walston-Felix	1977
IBM function points	1979
Jensen Seer	1979
COCOMO (Constructive Cost Model)	1981
SoftCost-R	1981
Estimacs	1983
SEER-SEM	1988
SPQR Checkpoint	1985
KnowledgePlan	1997

the practice of parametric software estimation than anyone in the industry, bringing the science of parametric estimating out of obscurity. His early work at TRW, followed by the COCOMO and the COCOMO II models and the many offshoots, have changed the industry forever.

Prof. Boehm's focus on software costs changed the direction of computing overall, not just estimating, as he predicted in 1972 that software would be the largest schedule and cost driver (overrunning hardware costs) and that there should be a focus on software rather than more powerful hardware (Whitaker, 1993).

Of course, estimating's purpose extends far beyond just knowing what something will cost. One can also use estimates to support business decisions, choose the most cost effective approaches, cancel unaffordable projects before they begin, and provide a basis for an achievable plan. In that regard, Prof. Boehm's value-based software and system engineering work focuses on using cost analysis as one side of the scale and value as the other side. The approach helps software become part of the overall contribution to the business rather than a cost and helps ensure the most reasonable choices for projects are made.

Galorath recalled being asked in the early '70s when a software product would be completed. His honest answer, "We don't know—we will tell you when we are finished building it" reflects the state of the practice in those days. Schedules were made only by heroic efforts of dedicated developers. Functionality often suffered; testing was often not complete, but when a date was critical programmers did the best they could. Additionally, Galorath recalled the first project he was asked to estimate in 1976. The estimate was not what management wanted to hear, and the project was cancelled because a schedule cut of a third was not offered up. It was then outsourced to a company that promised an eight-times improvement in cost and schedules and ended up delivering at over twice the schedule and three times the cost of the original, in-house estimate. Viable estimating might have averted this disaster.

Author Galorath also recalled applying then-current parametric estimation models to troubled software projects in the early '80s. The hardware engineers reigned supreme, and software developers were considered second-class citizens. In this project, the hardware people had arbitrarily sized the memory in a new product. The software project was failing due to lack of memory. Yet the hardware people refused to change the memory size, demanding the software people "do their job and quit making excuses." Using early

parametric estimation technology, management was able to see the value of spending a little more on hardware to gain substantial cost and schedule savings in software. Once this hardware change was implemented, the project completed, shipped, and changed the particular industry with major technical step forward.

Prof. Boehm's work impacted all of us in the cost modeling community. For example, even though the original Jensen Seer software cost model has roots separate from COCOMO, when Prof. Boehm's software economics book (Boehm, 1981) was released, several Seer parameters were rescaled to be consistent with COCOMO. At this same time, Reifer and Galorath had just recently completed a study recommending parametric estimating for early estimation of software costs (Galorath & Evans, 2006). Prof. Boehm's work became instantly more visible to the community helping to generate acceptance of the technology.

Estimation is the catalyst for many measurement initiatives for both systems and processes. Prof. Boehm's reach has certainly been into software system measurement and processes. Better estimation required better data, which in turn required better measurement. Better software processes helped repeatability. In the forward to Galorath and Evans (2006), Prof. Boehm proposed that just running the parametric model is the small tip of the iceberg—that estimation is about planning. The key points he proposed are:

1. *Identifying what is being estimated and why:* One early cost modeler's answer to questions asking whether the model estimates included costs of management or quality assurance was, "What would you like the estimates to include?" This is not a strong confidence builder.
2. *Defining the project's requirements and design as well as possible.* If you don't know whether a product function will be fulfilled by new, modified, or commercial software, your estimates can be way off.
3. *Using several perspectives to estimate software size, cost, and schedule.* Otherwise, there is no way to tell whether your estimates are reasonably accurate or not.
4. *Identifying ranges of uncertainty in the project parameters.* This enables techniques such as Monte Carlo analysis to determine the likelihood of finishing within a given budget or schedule. Just using a "most likely" point estimate will overrun roughly half of the time.
5. *Performing a business case relating estimated costs to estimated benefits and return on investment.* Otherwise, scarce resources are likely to be spent on low-payoff capabilities.
6. *Negotiation of tradeoffs among cost, schedule, quality, performance, and functionality.* Optimizing on one of these parameters at the expense of the others has been the source of many failed projects.
7. *Matching desired capabilities to available budgets, schedules, and skilled personnel.* Neglecting this activity has been the source of many project overruns.
8. *Tracking not only cost and progress with respect to original cost and schedule estimates, but also changes in cost driver parameters.* Tracking to obsolete estimates has been the source of many painful surprises.

Untold billions of dollars and careers have been saved by viable, repeatable estimating. And Prof. Boehm is largely to thank for that.

Systems and Software Engineering Spiral Life Cycle Model

In 1970, Winston Royce (1970) described key flaws in the sequential development processes in use at the time. This process, later referred to as the "waterfall model," was risky and invited failure due to the required requirements and design rework often discovered

late on large, complex projects. Even with feedback loops to handle problems, there was a tendency to "salvage" as much as possible of the design already done. A better approach was needed that focused on identifying key risks and working to minimize or eliminate these risks up front, thereby reducing the need for late rework that often led to failed system development efforts. Winston suggested in his paper that if one planned to build the system twice (the first time to learn, the second time to develop an effective system for the customer using what we learned), the results would be much more satisfactory. This "build it twice" approach led to prototyping, where the first version was a prototype and the second version was the actual delivered system.

Prof. Boehm formalized much of this in the mid 1980s in the spiral development model (Boehm, 1986) and tied the prototyping efforts to identified high risks. The spiral development model could help programs navigate through immature technologies and other high-risk elements in programs and, in conjuction with cost estimation tools, could be used to better manage large, complex software-intensive system development projects. However, some of the largest, most complex systems in the United States were developed by the Department of Defense (DoD), and the spiral development model was inconsistent with the DoD system and software development standards that were based upon the waterfall process. Prof. Boehm took on the challenge to demonstrate to the DoD how they could be more successful by using the spiral development process on large, complex, risky development programs. In 2002, the DoD stated that software development should follow a spiral development process (DoD, 2002). However, as Prof. Boehm observed projects using the spiral model, he found that they often misinterpreted the model and were soon "spiraling out of control" and sometimes "augering in" with little to show for their efforts.

Further analysis showed that while many development organizations wanted to successfully develop systems, the acquisition processes continued to stifle the development process by expecting developers to provide more functionality with lower costs and shorter schedules. These constraints encouraged overly optimistic bids which led to early program problems and little leverage to do anything about them until the developer failed to deliver the system several years later. Prof. Boehm realized that further acquisition reform was needed and began to add more rigor to the spiral development model. This became the Incremental Commitment Spiral Model (ICSM) (Boehm, Lane, Koolmanojwong & Turner, 2014). The ICSM is based on principles employed by successful programs and incremental development phases with rigorous feasibility assessments between phases. This process allows the acquisition community to discontinue non-viable or obsolete programs and reprioritize and fund more promising programs.

As is Prof. Boehm's way, he started developing the ICSM slowly, analyzing both successful and failed programs, assembling the ICSM framework with encouragement from others, including the National Academies (Pew & Mavor, 2007), working through the details in the classroom environment with his student's projects, and taking his message to conferences and workshops around the world. As a result, many of us are now familiar with his some of his favorite ICSM gambling[1] and dating[2] analogies. In addition, through this ICSM socialization process, he found potential early adopters which allowed him to capture and incorporate lessons learned into new iterations of his ICSM papers and tutorials. Out of these efforts came better insights into conducting meaningful feasibility assessments, competitive prototyping to pursue more than one approach for large, risky systems, guidance on how much to invest in early prototypes, the investment "sweetspots," and the IDPD to better estimate incremental development effort.

Prof. Boehm's Academic Family Tree: From 1994 to 2014

Just as Prof. Boehm came from a long lineage of mathematicians, he formed a family of his own by advising dozens of doctoral students. In 1992, Prof. Boehm arrived at the University

of Southern California Computer Science Department as the TRW Professor of Software Engineering. Since that time, he has advised 36 doctoral students, as shown in the family tree in Figure 1.

Prof. Boehm shared a number of success strategies with his students that are worth mentioning in his presentation entitled "Getting Published," given at Loughborough University (United Kingdom) to a doctoral student workshop (Boehm, 2009):

1. *Discover your strengths and build on them*. Prof. Boehm suggested that students should recognize what they are good at and leverage those skills to support research. For example, strength in statistics may enable a research path that is heavily focused on quantitative analysis of large data sets. Alternatively, capabilities and interests in modeling can open doors in areas where simulation is an adequate methodology. Achieving the right fit between a researcher's strengths and dissertation topic is more likely to lead to high-impact research.

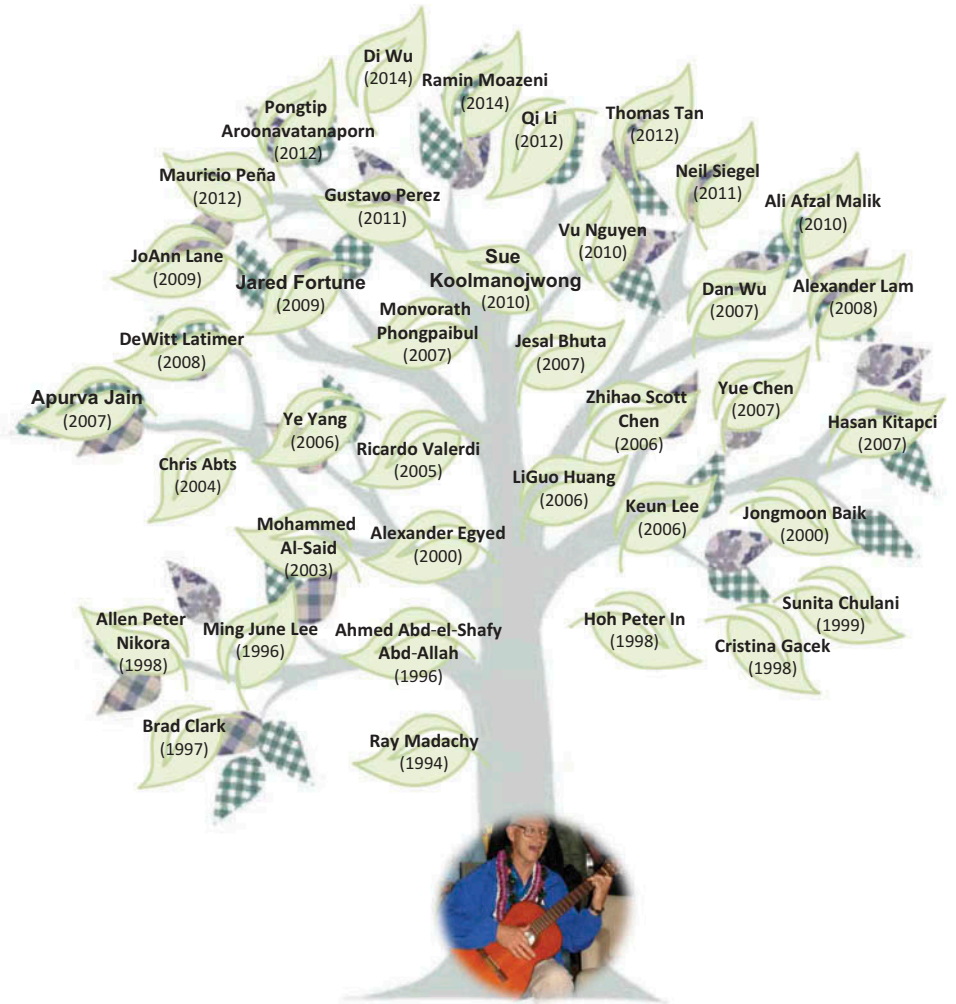

FIGURE 1 Prof. Boehm family tree.

TABLE 3 Prof. Boehm's academic family tree includes 36 doctoral alumni

Name	Year	Ph.D. Research Area	Employer
Ray Madachy	1994	Software project dynamics model for project cost, schedule, and risk assessment	Naval Postgraduate School
Ahmed Abd-Allah	1996	Composing heterogeneous software architectures	Computing Innovation
Ming June Lee	1996	Foundations of the WinWin requirements negotiation system	Synopsis
Brad Clark	1997	Effects of software process maturity on software development effort	Software Metrics
Cristina Gacek	1998	Detecting architectural mismatches during systems composition	University of Newcastle
Ho (Peter) In	1998	Conflict identification and resolution for software attribute requirements	Korea University
Allen Nikora	1998	Software system defect content prediction from development process and product characteristics	NASA JPL
Sunita Chulani	1999	Bayesian analysis of software costs and quality models	Cisco
Jongmoon Baik	2000	Effects of CASE tools on software development effort	Korea Advanced Institute of Science and Technology
Alex Egyed	2000	Heterogeneous view integration and its automation	Johannes Kepler University
Mohammed Al-Said	2003	Detecting model clashes during software systems development	Saudi Aramco
Chris Abts	2004	Commercial Off the Shelf (COTS) extension to COCOMO	Booz Allen Hamilton
Ricardo Valerdi	2005	Constructive Systems Engineering Cost Model	University of Arizona
Scott (Zhihao) Chen	2006	Reduced-parameter modeling for cost estimation models	Motorola
LiGuo Huang	2006	Software quality analysis: A value-based approach	Southern Methodist University
Keun Lee	2006	Development and evaluation of value-based review methods	Samsung
Ye Yang	2006	Composable risk-driven processes for developing software systems from COTS Products	Stevens Institute of Technology
Jesal Bhuta	2007	Framework for intelligent assessment and resolution of COTS product incompatibilities	Infosys
Yue Chen	2007	Software security economics and threat modeling based on attack path analysis	Cisco
Apurva Jain	2007	Theory of value-based software engineering	Microsoft
Hasan Kitapci	2007	Formalizing informal stakeholder inputs using gap-bridging methods	Consultant

Monvorath Phongpaibul	2007	Experimental and analytical comparison between pair development and software development with Fagan's inspection	IBM
Danni Wu	2007	Security functional requirements analysis for developing secure software	Microsoft
Alex Lam	2008	Architecture and application of an autonomous robotic software engineering technology test bed	The Aerospace Corporation
DeWitt Latimer	2008	Effectiveness of engineering practices for the acquisition and employment of robotic systems	U.S. Air Force
Jared Fortune	2009	Estimating systems engineering reuse with the Constructive Systems Engineering Cost Model	The Aerospace Corporation
Jo Ann Lane	2009	Impacts of system of system management strategies on system of system capability engineering effort	USC
Sue Koolmanojwong	2010	The incremental commitment spiral model process patterns for rapid-fielding projects	IBM
Ali Afzal Malik	2010	Quantitative and qualitative analyses of requirements elaboration for early software size estimation	University of Lahore
Vu Nguyen	2010	Improved size and effort estimation models for software maintenance	Vietnam National University
Gustavo Perez	2011	Using metrics of scattering to assess software quality	MTM Tecnologia
Neil Siegel	2011	Organizing complex projects around critical skills and the mitigation of risks arising from system dynamic behavior	Northrop Grumman
Pongtip Aroonavatanaporn	2012	Shrinking the cone of uncertainty with continuous assessment for software team dynamics in design and development	Pharmacare Limited
Mauricio Peña	2012	Quantifying the impact of requirements volatility on systems engineering effort	Boeing
Qi Li	2012	Value-based, dependency-aware inspection and test prioritization	Yellowpages.com
Thomas Tan	2012	Domain-based effort distribution model for software cost estimation	Tangent Solutions
Ramin Moazeni	2014	Incremental development productivity decline	Oracle
Di Wu	2014	Wiki WinWin—A Wiki-based collaboration framework for rapid requirements negotiations	Oracle

2. *Attack the future*. Rather than working on yesterday's challenges, Prof. Boehm encouraged students to anticipate the challenges of the future such as user value/focus, human systems integration, off-the-shelf/legacy integration, distribution/mobility/globalization, emergence, and services. Many of these categories are elaborated in a visionary article about systems and software engineering trends (Boehm, 2006).
3. *Look for applications*. Despite the heavy emphasis on theoretical contributions, Prof. Boehm suggested that a doctoral dissertation should also have clear applications in a variety of contexts. Part of this involves finding out what users need the most and turning those needs into research opportunities.
4. *Strive for excellence and impact*. Prof. Boehm noted that characteristics of successful papers, in addition to rigor and relevance, include originality, clarity, and utility. Furthermore, researchers should aim to explain their work to people, especiallynon-researchers (Reddy, 2009). Prof. Boehm pointed to Hamming's (1986) suggestions that in order to do great research, you must have courage, emotional commitment, the ability to think and continue to think, to turn defects into assets, sometimes neglect things if you intend to get what you want done, and recognize that the steady application of effort with a little bit more work *intelligently applied* is what does it.

Prof. Boehm's students have all valued their time spent him and now carry on his research legacy in a variety of areas related to software economics in various positions in government, industry, and academia, as shown in Table 3.

Conclusion

The worlds of software engineering and systems engineering owe a huge debt of gratitude to Prof. Boehm. He has given software engineering respectability through his research contributions. He has grown the next generation of leaders in the engineering community, his innovations have saved billions of dollars, and he has advanced the cost estimation profession dramatically.

Notes

1. In the gambling analogy, incremental commitment is more like playing the game of poker than roulette. In poker, the players place a minimum bet or ante (initial system investment), receive some cards, then decide whether to bet more (additional investments) or fold. Round of betting (investments) can iterate many times. In roulette, you bet once (fully funded contract up front) and hope for the best when the wheel is spun.
2. In the dating analogy, ICSM is more like the traditional dating process where one dates for a while and if things go well, the couple decides to invest further in the relationship and go steady. After going steady for a while, if the couple decides that they want a further commitment, they decide to get married and have children. This dating process tends to work much better than the "date, have a child, marry in haste, and repent at leisure" process.

References

American Mathematical Society, Mathematics Genealogy Project, Retrieved from http://www.genealogy.ams.org/.

Boehm, B. W. (1981). *Software engineering economics*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Boehm, B. W. (1986). A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes*, *11*(4), 14–24.

Boehm, B. W. (2006). Some future trends and implications for systems and software engineering processes. *Systems Engineering*, *9*(1), 1–19.

Boehm, B. W. (2009). *Getting published*. Paper presented at the INCOSE Systems Engineering & Architecting Doctoral Student Network, Loughborough, UK.

Boehm, B. W., Lane, J., Koolmanojwong, S., & Turner, R. (2014). *The incremental commitment spiral model: Principles and practices for successful systems and software*. Upper Saddle River, NJ: Addison-Wesley Professional.

Boehm, B. W., & Valerdi, R. (2008). Achievements and challenges in software resource estimation. *IEEE Software*, *25*(5), 74–83.

Department of Defense [DoD]. (2002). *Operation of the Defense Acquisition System. DoDI 5000.2*, April 5, 2002, Washington, D.C.

Galorath, D., & Evans, M. (2006). *Software sizing, estimation and risk management, When performance is measured performance improves*. Danvers, MA: Auerbach Publications.

Hamming, R. (1986). *You and your research*. Paper presented at the Bell Communications Research Colloquium Seminar. Retrieved from http://www.cs.virginia.edu/~robins/YouAndYourResearch.html

Merton, R. K. (1965). *On the shoulders of giants: A Shandean postscript*. New York, NY: Free Press.

Pew, R. W., & Mavor, A. S. (2007). *Human-system integration in the system development process: A new look*. Washington, DC: National Academy Press.

Reddy, C. (2009). Scientist citizens. *Science, 323*(March 13), 1405.

Royce, W. W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON 26*, pp. 1–9.

Whitaker, W. (1993). Ada—The project: The DoD high order language working group. Retrieved from http://archive.adaic.com/pol-hist/history/holwg-93/holwg-93.htm

About the Authors

Jo Ann Lane is the systems engineering Co-Director of the University of Southern California Center for Systems and Software Engineering, leading research in the areas of software engineering, systems engineering, and system of systems engineering (SoSE). She received her PhD in systems engineering from the University of Southern California and her Master's in computer science from San Diego State University.

Daniel D. Galorath is the President and CEO of Galorath Incorporated and the chief architect of SEER-SEM, an algorithmic project management software application. He is a recognized expert in the fields of software estimation and sizing and the author of Software Sizing, Estimation, and Risk Management. He attended California State University where he completed both his undergraduate and graduate work. He graduated in 1980 with an MBA in management.

Ricardo Valerdi is an Associate Professor of Systems and Industrial Engineering at the University of Arizona. He is the inventor of the COSYSMO cost model and has over 100 publications in the areas of systems engineering and cost estimation. He obtained his Ph.D. from the University of Southern California.