

The Software Developer as Movie Icon

Warren Harrison



As a college professor, I often get an opportunity to speak with incoming freshmen who have decided to major in computer science. Virtually all these young people share a single attribute: they have no idea what a professional software developer does. This means that many students who pick this career will either be unsuccessful or, worse yet, successful at a career they'll hate until they retire. At the same time, many students who would find the profession enjoyable and be quite good at it might not give it a second thought.

And they won't learn much more about the business in their first few years in college, either. They'll probably get a good dose of writing simple C++ or Java programs, almost invariably as a solo effort and after the professor has given them a detailed set of specifications of how the assigned software is to behave. By the end of their second year, most students are certain the software development business is all about `while` loops, `if` statements, and being very, very clever and self-reliant. In this distorted view of our profession,

- The developer always has a complete spec (or if they don't, they can make it up as they go along).
- All development begins with a brand new program.

- If your program doesn't core-dump in response to a test case, its behavior is correct.
- No one else (with the exception of the professor, perhaps) ever looks at your code.
- The developer never, ever has to read someone else's code.

By the end of their fourth year, if they're lucky, the hopeful software developer will have started to catch on. Unfortunately, most will just view the team projects, class presentations, software life cycles, and ambiguous problem statements as "hoops they have to jump through" to graduate. I recall a meeting with a student who was having trouble working with her project team in a recent class. She actually asserted with some confidence that the problem she was having with her team members (not acknowledging her as the finest software architect in the land) wouldn't happen in industry. She was stunned when I explained that her teammates were going to graduate the same time she was and that even if she might not end up working with them, she would likely be working with someone much like them.

Delivering a dose of reality

I have recently begun trying to learn how to better prepare students for the reality of software development. This doesn't mean looking for more realistic projects or figuring out how to move code inspection exercises into freshman Introduction to Computer Science classes. If young people had a better un-

DEPARTMENT EDITORS

Bookshelf: Warren Keuffel,
wkeuffel@computer.org

Construction: Andy Hunt and Dave Thomas,
Pragmatic Programmers,
{Andy, Dave}@pragmaticprogrammer.com

Design: Martin Fowler, ThoughtWorks,
fowler@acm.org

Loyal Opposition: Robert Glass, Computing Trends,
rglass@indiana.edu

Manager: Don Reifer, Reifer Consultants,
d.reifer@ieee.org

Quality Time: Jeffrey Voas, Cigital,
voas@cigital.com

STAFF

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

Group Managing Editor
Crystal Chweh

Associate Editors
Shani Murray and Dennis Taylor

Editorial Assistants
Rebecca Deuel and Ty Manuel

Magazine Assistant
Pauline Hosillos, software@computer.org

Art Director
Toni Van Buskirk

Cover Illustration
Dirk Hagner

Technical Illustrator
Alex Torres

Production Assistant
Monette Velasco

Production Artists
Carmen Flores-Garvey and Larry Bauer

Executive Director
David Hennage

Publisher
Angela Burgess

Assistant Publisher
Dick Price

Membership/Circulation Marketing Manager
Georgann Carter

Advertising Assistant
Debbie Sims

CONTRIBUTING EDITORS

Kirk Kroecker, Christine Miller, Keri Schreiner, and
Joan Taylor

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.

derstanding of what professional software development is really like, those with a propensity for working in teams, coding to a specification, and following rigorous, well-defined processes would more likely study software development, and those not so predisposed wouldn't.

As my sons get closer to high school graduation, I have noticed that they and their friends seem to know a lot about some professions (software development is not one of them, in spite of their father), and I am trying to understand how they acquired this knowledge. As far as I can tell, the best-understood career path for young adults (at least in North America) is a toss-up between law enforcement and firefighting. They know the vernacular, they can describe the equipment, and they can immediately recognize the agency initials. I suspect the reason is that young people these days tend to get a lot of their information from television and movies. While they end up learning some incorrect (or at least distorted) information by watching "reality" TV, most young people have a pretty fair idea of what police officers or firefighters do, the environment in which they work, and the organizational structure within which they operate.

Hollywood lessons

On the other hand, most software developer hopefuls don't have a clue about what they will be spending the rest of their lives doing. And the information they do receive from movies about software developers is consistently inaccurate. Because of this, I have begun a semi-serious study of how Hollywood portrays software developers, analyzing a number of classics in which computers and software play major roles in the plot. Disney's *TRON* (1982), *WarGames* (1983), and *The Net* (1995) are representative of the way movies portray computer folk. The image that emerges is not a pretty one.

In *TRON*, Jeff Bridges is a gifted programmer who singlehandedly writes several computer games that

are subsequently stolen by David Warner. These titles become huge hits in the marketplace and propel Warner to become executive senior vice president. Angry that his work has been stolen, Bridges quits his job and retreats into a video arcade, from which he emerges only to try and break into his previous employer's computers. While Bridges portrays a cooler than normal computer geek, he nevertheless manages to give the impression that his life revolves around computer games. Eventually, he gets electronically sucked into the mother of all computers and, as the saying goes, the rest is history.

In *WarGames*, Matthew Broderick is a high school nerd with few friends. He does somehow end up with a girlfriend (Ally Sheedy), but she is just as nerdy as he is. In his spare time (since he doesn't spend much effort on his school work, he appears to have plenty of it), he hacks into random computers using an automatic phone dialer that calls random numbers until he connects to a modem. He finally hacks into the NORAD Defense Department War Computer, which has complete control over the US nuclear arsenal. (In the movie, this computer is called WOPR. I have been told that NORAD's real system in the 1970s was called BURGR.) Naturally, the software controlling every US missile was written by a single person (John Wood), who has retreated to a cabin in the Pacific Northwest where he lives alone and avoids visitors.

In *The Net*, Sandra Bullock plays a software developer who works from home and spends her free time in online chat sessions. She is so isolated from in-person social interactions that when her identity is stolen, she can't find a single individual that can physically identify her. The fact that her personality is better developed as the plot unfolds and she becomes less of a comic book character than the software experts in the first two films makes her portrayal a bit more unsettling.

There are a dozen other films that reinforce this image of software de-

velopers as brilliant yet socially awkward individuals who code on the fly, singlehandedly develop ultracomplex systems, and aren't above breaking into a computer or two. Not surprisingly, this misconception attracts to computer science many brilliant, socially awkward young people who code on the fly, think they can singlehandedly develop ultracomplex systems, and are interested in breaking into computers.

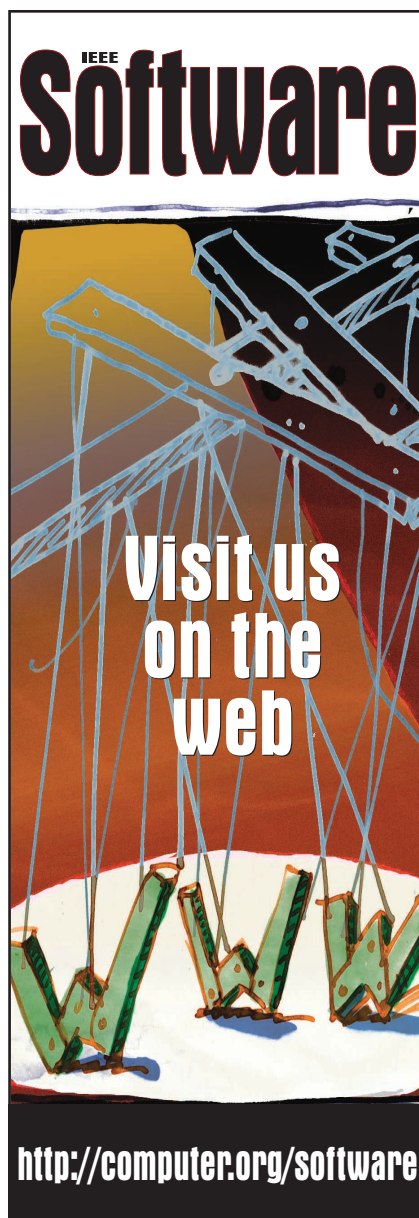
Ironically, as a college professor, I have the opportunity to meet a large number of employers of software developers. Invariably they are looking for graduates who socialize well in groups, are team players, are articulate, and are able to give coherent oral presentations. Although they obviously expect some technical ability, overwhelmingly the traits that most distinguish new graduates are communication abilities and the willingness to be a team contributor. This is exactly the opposite of movies' portrayal of software developers.

The film critic Roger Ebert once wrote that it was a grave mistake for professionals to see movies about their trade because it is impossible to suspend disbelief when you see your trade misrepresented. But the way software developers are portrayed in the cinema goes beyond simple misrepresentation. Rather, it influences the way young people—potential future software developers—think about the profession. It can encourage students who will never be happy working on programming teams, dealing with customers, or working through a well-defined software process to seek software development careers. More importantly, it can discourage those who would welcome such an environment but mistakenly fear spending their career stuck in a cubicle hunched over a computer monitor.

Of course, when all is said and done, these movies and others like them are immensely entertaining, and as long as we can keep young people from thinking that they even

remotely represent the reality of software development, I suppose they really cannot be too harmful.

I'd love to hear your opinions about the impact of the cinema and television on new software developers' expectations, as well as learn of any films that do a better job of portraying our profession. Please write to me at Warren.Harrison@computer.org. ☉



EDITOR IN CHIEF:

Warren Harrison
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314
warren.harrison@computer.org

EDITOR IN CHIEF EMERITUS:

Steve McConnell, Construx Software

ASSOCIATE EDITORS IN CHIEF

Design: Maarten Boasson, Quaerendo Invenietis
boasson@quaerendo.com

Construction: Terry Bollinger, Mitre Corp.
terry@mitre.org

Requirements: Christof Ebert, Alcatel Telecom
christof.ebert@alcatel.com

Management: Ann Miller, University of Missouri, Rolla
miller@ece.umar.edu

Quality: Jeffrey Voas, Cigital
jvoas@cigital.com

Experience Reports: Wolfgang Strigel,
Software Productivity Center; strigel@spc.ca

EDITORIAL BOARD

Don Bagert, Rose-Hulman Institute of Technology
Nancy Eickelmann, Motorola Labs
Richard Fairley, Oregon Graduate Institute
Martin Fowler, ThoughtWorks
Robert Glass, Computing Trends
Jane Hayes, University of Kentucky
Andy Hunt, Pragmatic Programmers
Warren Keuffel, independent consultant
Brian Lawrence, Coyote Valley Software
Karen Mackey, Cisco Systems
Deependra Moitra, Lucent Technologies, India
Don Reifer, Reifer Consultants
Suzanne Robertson, Atlantic Systems Guild
Dave Thomas, Pragmatic Programmers

INDUSTRY ADVISORY BOARD

Robert Cochran, Catalyst Software (chair)
Annie Kuntzmann-Combelles, Q-Labs
Enrique Draier, PSINet
Eric Horvitz, Microsoft Research
David Hsiao, Cisco Systems
Takaya Ishida, Mitsubishi Electric Corp.
Dehua Ju, ASTI Shanghai
Donna Kaspersen, Science Applications International
Pavle Knaflic, Hermes SoftLab
Wojtek Kozaczynski, Rational Software Corp.
Tomoo Matsubara, Matsubara Consulting
Masao Matsumoto, Univ. of Tsukuba
Dorothy McKinney, Lockheed Martin Space Systems
Stephen Mellor, Project Technology
Susan Mickel, AgileTV
Dave Moore, Vulcan Northwest
Melissa Murphy, Sandia National Laboratories
Kiyoh Nakamura, Fujitsu
Grant Rule, Software Measurement Services
Girish Seshagiri, Advanced Information Services
Chandra Shekaran, Microsoft
Martyn Thomas, Praxis
Rob Thomsett, The Thomsett Company
John Vu, The Boeing Company
Simon Wright, Integrated Chipware
Tsuneo Yamaura, Hitachi Software Engineering

MAGAZINE OPERATIONS COMMITTEE

George Cybenko (chair), Thomas J. Bergin, Pradip Bose, Doris L. Carver, John Dill, Frank E. Ferrante, Robert Fillman, Forouzan Golshani, Rajesh Gupta, Warren Harrison, Mahadev Satyanarayanan, Nigel Shadbolt, Francis Sullivan

PUBLICATIONS BOARD

Rangachar Kasturi (chair), Jean Bacon, Mark Christensen, George Cybenko, Thomas Keefe, Richard A. Kemmerer, Gabriella Sannitti di Baja, Steven L. Tanimoto, Anand Tripathi