Agile principles and ethical conduct

Ken H. Judy
Simon and Schuster
kjudy@computer.org

Abstract

Software practitioners experience pressure to compromise their work and their reasonable care for others. Even as software becomes more beneficial, pervasive, and interconnected, the potential for unintended harm grows. Agile Software Development is an approach to building software systems that embodies a set of declared core principles. How do these principles align to an ethical standard of conduct? This paper attempts from an Agile Practitioner's perspective to compare and contrast Agile Principles with other approaches to Software Ethics. It identifies areas of strong resonance and gaps that exist between the stated Agile Principles and an explicit software code of ethical conduct.

1. The potential for harm and for benefit

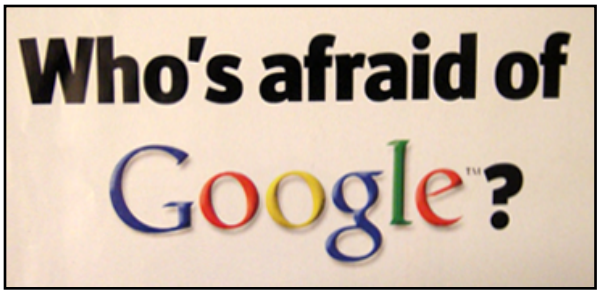

Figure 1: Economist Cover, September 2007

As our identities and activities become wed to software systems, developers have begun to affect our lives in ways and to degrees we hadn't anticipated.

"In the discreet world of computing, there is no meaningful metric in which small change and small effects go hand in hand." - (Dijkstra 1989. p. 1400) … the normally predictable linkage between acts and their effects is severely skewed by the infusion of computing technology." [1]

This is due in part to miniaturization, global interconnectivity and commoditization. More and more people have access to devices with more and more computational power.

"Thus while [integrated circuits] are a primary driver of complexity in modern day objects, they also enable the ability to shrink a frighteningly complex machine to the size of a cute little gum-drop… There is no turning back to the age when large objects were complex and small objects were simple." [2]

These underlying advances spur evolution in the software industry with new languages and frameworks and lower thresholds to creating complex systems. The worldwide software developer population is expected to grow to 19.5 million by 2010 from 14.5 million in 2007. [3]

The growing influence of software in our society is also driven by our transition from an industrial to a service economy. According to the US State Department, in 2006 67.8% of US GDP came from the services sector. The US maintains a $79.8B trade surplus in services while carrying an $800B trade deficit in manufactured goods. [4]

So, as providing services through machine interfaces has become more achievable, our lives and our livelihoods have increasingly begun to revolve around those services.

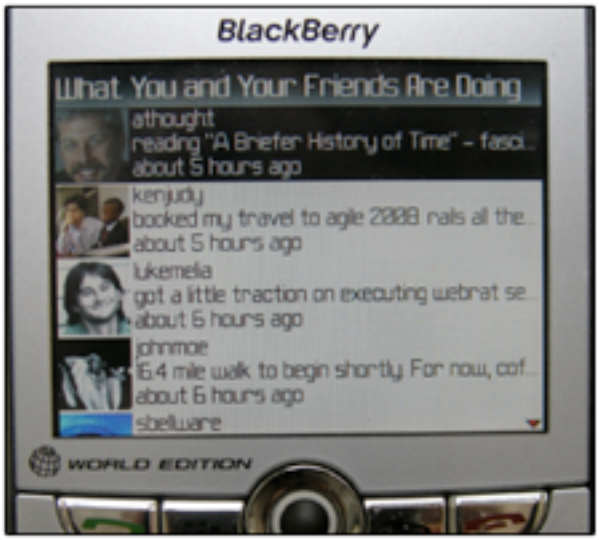

Figure 2. Identity on a Mobile Device

This pervasiveness leads to indirect chains of cause and effect among systems. Dependencies may be

intrinsic such as the linking of a retail shopping account to a credit card. Dependencies may be accidental. Credentials revealed on an entertainment website may unlock a bank account on another site.

"The cause of many such surprises seems clear: The systems involved are complex, involving interaction among and feedback between many parts. Any changes to such a system will cascade in ways that are difficult to predict; this is especially true when human actions are involved." [5]

This creates opportunity for criminality by software developers and others who exploit software systems.

11/9/2007 LOS ANGELES (Reuters) - "A Los Angeles man on Friday admitted infecting 250,000 computers and stealing the identities of thousands of people by wiretapping their communications and accessing their bank accounts." [6]

"In 2006, 36% of all complaints to the FTC were related to identity theft (246K incidents) another 11% to other online activities." [7]

It also creates the potential for unintended harm by well-intentioned software practitioners.

"Every week hundreds of vulnerabilities are being reported in web applications, and are being actively exploited. The number of attempted attacks every day for some of the large web hosting farms range from hundreds of thousands to even millions." - SANS Institute [8]

"Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion annually, or about 0.6 percent of the gross domestic product" - 2002 NIST Study [9]

"The Information and Communications Technology sector accounts for nearly 2% of global greenhouse gas emissions, and that data centres account for 23% of the direct footprint of the sector." -- Gartner [10]

Figure 3. Data Center

In technology, size is no longer a predictable measure of consequence. Small contains great expressive power. Not only small tools but mundane and seemingly insignificant decisions about their implementation can have tremendous consequences.

Software developers ought to be concerned with people and the world around them. As human beings, they should be determined to do more good in their lives than harm. Software developers need to consider the potential consequences of day to day decisions. This consideration is the domain of software ethics.

2. Agile software development

Agile software development encompasses several software project management and execution methodologies which evolved largely independently over the last twenty years. The originators and other advocates of these methodologies self-identify their practices as "agile" and are loosely associated through member organizations such as the Agile Alliance.

Agile software development incorporates a wide range of influences from pre-existing iterative and evolutionary development methodologies, empirical process control, games theory, lean product development, and learning gleaned from highly productive software development teams.

Some prominent Agile methodologies include:

Adaptive Software Development ASD (Jim Highsmith) - incorporates experience in RAD and the fundamental view of software development groups as complex adaptive systems. [11]

Crystal (Alistair Cockburn) - an array of approaches that add ceremony as criticality or scale increases. Emphasizes development as a cooperative game. [12]

Extreme Programming XP (Kent Beck, Ward Cunningham, Ron Jeffries) - primarily a set of mutually supporting coding practices based on direct experience with the first XP team at DaimlerChrysler. [site] [13]

Lean Software Development (Mary and Tom Poppendieck) - applies lean principles from the Toyota product system to software development. [14]

Scrum (Ken Schwaber, Jeff Sutherland, Mike Beedle) - a management and control process influenced by academic research on Japanese product development and empirical process control.[15]

3. Agile software development and the mushy stuff of values and culture

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value: Individuals and interactions over processes and tools. Working software over comprehensive documentation. Customer collaboration over contract negotiation. Responding to change over following a plan." [16]

What the originators of Agile practices held in common was a set of values they jointly published as

the *Manifesto for Agile Software Development*. The manifesto consists of a preamble and a list of twelve principles.

Figure 4: An agile development team

"At the core, I believe Agile Methodologists are really about 'mushy' stuff about delivering good products to customers by operating in an environment that does more than talk about 'people as our most important asset' but actually 'acts' as if people were the most important, and lose the word 'asset'. So in the final analysis, the meteoric rise of interest in and sometimes tremendous criticism of Agile Methodologies is about the mushy stuff of values and culture." -- Jim Highsmith [17]

4. The principles behind the Agile Manifesto[18]

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

5. Existing approaches to software ethics

Before discussing how Agile principles inform ethical decision making, it would be valuable to briefly review other systems of ethical thought in software. In particular, the craft model and the engineering approach to software ethics.

5.1. The craft guild and reputation

Advocates of the craft model of software development believe efficacy and quality arise from mastery and that the primary mode of learning is hands on supervised practice.

"What matters is that the people working in software development be skilled practitioners of their craft and that they are continually working to hone and improve their skills." [19]

Under a craft system, learning is transferred hands on as one ascends through mentorship and experience from apprentice to journeyman to master craftsperson.

For this model to exist in the typical software organization, radical changes need to occur in the industry. For example, the economics would have to change. Developers with few years of experience, i.e. apprentices, would need to be paid significantly less so that there would be fewer novices entering the field. This would create more appropriate ratios to allow mentoring from journeymen developers.

Experience would need to garner greater respect in the workplace. Developers with a decade or more of experience and mastery of craft would need to be retained in greater numbers. They would have to be paid significantly more and would need the authority required to build craft shops around themselves. Software suppliers would need to be selected for work based on personal recommendations and the reputations of the master craftspeople who head them.

If such changes were pervasive, the impetus for ethical action would fall on the need to protect one's reputation. "Peer recognition and recommendations are the route to better software. When one developer recommends another, he is putting his own reputation on the line." [20]

The problem with using the craft model as a guide for ethical behavior is that while reputation can be a strong incentive for personal standards it does not in

and of itself inform what those standards should be nor does it expand the area of concern to broader stakeholders who, while affected by a developer's decisions, are in no position to assign blame or affect reputation.

5.2. Plan-driven software ethics

A more systematic approach to addressing the moral dimensions of a practice is to professionalize. In the mid-nineteenth century, partly in response to the catastrophic failures of iron railway bridges, engineering "began to apply the scientific method to structural problems, it moved away from purely aesthetic considerations and separated itself from architecture" [21]

Thus began what Samual Florman calls "the golden age of engineering".

"Before 1850 there had been fine engineers and many outstanding engineering works. But engineering itself had been rather a craft than a profession, relying more on common sense and time-honored experience than on the application of scientific principles, and lacking those essentials of true professionalism -- professional schools and professional societies." [22]

In fact many advocates of this approach consider software development an emerging engineering profession. However, this question can be taken separately from a consideration of software development as profession.

A profession has requirements for learning and training, a code of ethics imposing higher standards *than normally tolerated in the marketplace*, a disciplinary system for those breaching the code, a primary emphasis on social responsibility, and licensing. [23]

A code of ethics "should instruct practitioners about the standards society expects them to meet, about what their peers should strive for, and about what to expect of one another. In addition, the code should also inform the public about the responsibilities that are important to the profession" [24]

Enforcing a code of ethics can serve and protect the public, provide guidance and inspiration, help practitioners arrive at shared standards, provide moral support and external validation for courageous decisions, educate and engender mutual understanding, provide deterrence and discipline and contribute to a discipline's image in the larger society. [25]

Software development has many existing codes of ethics. The IEEE has "The Software Engineering Code of Ethics and Professional Practice". The ACM has a "Code of Ethics and Professional Conduct". The British Computer Society has a "Society Code of Conduct". The Australian Computer Society has a "Code of Ethics".

At their best, codes of ethics introduce consideration for a broad set of stakeholders some of whom are present in the software development lifecycle: ourselves, co-workers, customers and employers. A code of ethics also advocates for stakeholders who are rarely present in day to day decisions: peers in the industry, end users, the reputation of software developers in society, the general public, and systems affected by the aggregate results of our industry such as the global environment.

Codes of ethics also delineate possible dilemmas a practitioner might encounter such as informed consent, safety and welfare, data integrity and representation, trade secrets, bribery, conflict of interest, accountability, and fairness. [26]

Consideration of these additional factors does set a "higher standard than normally tolerated in the marketplace."

However, codes of ethics are in the end only artifacts. Like all other attempts to codify essential complexity they are in danger "abstracting away the essence of the problem".

One danger is that a code of ethics will over-prescribe the software development lifecycle. For example, requiring upfront completeness in software design and specification.

"Software engineers shall commit themselves to making the analysis, specification, design, development, testing, and maintenance of software a …respected profession. Software engineers shall approve software only if they have a well-founded belief that it… meets specifications…" [27]

As methodologies for software development continue to evolve, such language may quickly become archaic or controversial.

Another danger is that while no document can capture the circumstances of even a small set of specific ethical dilemmas, a formally recognized and incomplete code lends itself to abuse and can be used to actually justify unethical action.

"Whatever the background, individuals may well be aware that acts they are about to perform do not meet the standards that society will consider to be appropriate for moral behavior. An individual my nonetheless, feel impelled to act in the way that society will judge immoral by the more immediate pressures... In such circumstances, finding what looks like a moral code that *does not condemn,* the course of action in question can come as a great relief." [28]

Therefore, a code of ethics for software development must acknowledge it's incompleteness. It must serve as a reminder of an ongoing conversation within the software development community and with our broad set of stakeholders. It must be a living document evolving to address changes in our field.

The agile approach to documentation is to think of it as "a reminder to have a conversation with

stakeholders" [29]. A written code can help remind practitioners to encompass a larger set of concerns but it is participating in an ongoing conversation among their peers, with ethicists, and with those affected by software that will help developers address the specific day to day dilemmas of software ethics.

6. Ethical gaps in the agile worldview

The IEEE Computer Society and Association for Computing Machinery *Software Engineering Code of Ethics and Professional Practice* identifies eight areas of concern: the public, the client and employer, the product, judgement, management, profession, colleagues, and self. Within each of those areas they identify 5-15 detailed principles. These areas of concern represent the IEEE-CS and ACM's attempt to capture the full range of software engineering ethical concerns.

The Agile Manifesto consists of three four values which are embodied in twelve principles. A comparison between these twelve Agile principles and the IEEE-CS/ACM eight areas of ethical concern suggests gaps and overlaps between these two ethical world views.

This, of course, does not imply that the original authors or current practitioners of Agile Software Development lack these values but simply that the original authors chose not to express them as part of their shared agenda.

However, as documentation is a reminder to have a conversation with stakeholders, gaps in the Agile principles suggest stakeholders who may be under-served, or at least, under-represented in the literature, discourse and practice of Agile Software Development.

As a concession to dissensus between the drafters of both documents, requirements for documentation artifacts are omitted. This is because consideration of the value of artifacts is redundant to the vast majority of Agile literature and is beyond the scope of this paper.

The first table focuses on how the stated Agile Principles shore up an ethical world-view. It is organized around the IEEE-CS/ACM's eight areas of concern. The second column lists which of the twelve Agile principles directly benefit that area of concern.

Table 1. Areas of Concern/Agile Principles

IEEE-CS/ACM Area of Concern	Supporting Agile Manifesto Principle
Public	
Client/Employer	continuous delivery welcome changing requirements working software work together
Product	continuous delivery welcome changing requirements working software technical excellence simplicity
Judgement	technical excellence simplicity self-organizing teams reflection
Management	work together motivated individuals face-to-face conversation sustainable development self-organizing teams reflection
Profession	sustainable development technical excellence self-organizing teams reflection
Colleagues	work together motivated individuals face-to-face conversation sustainable development self-organizing teams reflection
Self	sustainable development technical excellence self-organizing teams reflection

The second table focuses on ethical gaps in the Agile world-view as contained in the Agile Manifesto. This table is also organized around the IEEE-CS/ ACM's eight areas of concern. The second column focuses on the detailed principles the IEEE-CS/ACM. In particular, which of these detailed principles have no direct analog in the twelve principles of the Agile Manifesto.

Table 2. Gaps in Stated Agile principles

IEEE-CS/ACM	Agile Manifesto
Public	Protect the public good. Disclose potential danger to the user, the public, or the environment Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that can diminish access to the benefits of software. Volunteer
Client/Employer	
Product	
Judgement	Temper all technical judgments with human values. Do not abet conflicts of interest.
Management	
Profession	Avoid associations with unethical businesses and organizations Report significant ethical violations
Colleagues	Encourage colleagues to adhere to ethical conduct.
Self	Improve their knowledge of relevant standards and the law Avoid prejudice

7. Agile principles are a compelling but incomplete set of ethical concerns

The Agile Principles represent a commitment to delivering business value, software quality, honesty, introspection, continuous improvement, humane work environments, empowered workers and customer collaboration.

Their strength is with regard to behaviors the provide direct value to the customer, a sustainable pace, and team and individual excellence..

Agile Principles are silent on the responsibility of the software developer to the general well-being. This includes both things developer should avoid such as participating in actions that benefit customers and employers but potentially harm un-empowered and distant stakeholders. It also omits things developers should do such as volunteering, encouraging fair distribution of computing resources, and being aware of developers standing in law.

Finally, Agile Principles don't consider software developer's responsibility for the conduct of software developers outside their immediate collaborative team.

8. Expanding agile practice to consider broader ethical concerns

The author of this paper has no clear formula for leveraging agile practices to build higher standards of conduct in software development organizations.

However, several courses of action and further investigation should be pursued.

• Research should be done on ethical dilemmas specific to Agile Software Development and on the contribution of Agile to ethical software development.

• Thought leaders in the agile community should begin to discuss Agile values and principles as ethics.

• Thought leaders in the agile community should acquire a more sophisticated understanding of the current investigations and theories of ethics and psychological and cognitive aspects of moral thinking.

• Professional literature and active online community should expand the range of concerns to encompass more than business value, efficacy, and quality and should explicitly include obligations to end users, to society, and to the field of software development.

• The agile community should take more responsibility for the actions of peers and champion examples of ethical behavior and censure examples of unethical behavior in our midst.

• Agile practitioners should engage in a conversation with ethicists in the larger software development industry, in engineering and in academia. Agile should continue the tradition of learning

practices from other industries. Agile should also receive attention and credit for the contribution it is making to ethical, principled software development.

• Agile practitioners with assistance from the larger software development and academic communities should adapt existing forums, techniques and language to safely discuss ongoing and real ethical dilemmas in a way that does not violate employee agreements or unnecessarily endanger job security.

• Agile training curricula should include a survey of ethical concerns such as informed consent. To be agile should mean to be an informed an ethical practitioner.

• Agile practitioners should apply existing agile practices of retrospection, user-centered design, participation of stakeholders to elicit broader ethical implications of decisions and projects in the field and document and share learning through experience reports.

9. Conclusion

"We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct" [30]

We, software practitioners experience pressure to compromise our work and our reasonable care for others.

At the same time, the economy has become more and more bound to services delivered through software interfaces. Software systems are proliferating. Complexity is increasing. Inter-dependancy is increasing. People's reliance on software systems is increasing.

As software becomes more beneficial, more pervasive, and interconnected, our potential to harm grows.

Agile practices are designed to navigate essential complexity. Their growing rate of adoption is based upon a founding set of ethical concerns, "The mushy stuff of values and culture." The agile community itself provides a vital resource of seasoned peers with shared values.

By their very nature, agile practices founded within a set of ethical principles makes a contribution to conduct in our field. This despite Agile Principles providing an incomplete ethical system. The conversation on ethical dilemmas is largely absent from an Agile context where they do not directly affect business value or teams.

Nonetheless, by expanding the scope to encompass a broader range of ethical concerns, applying relevant Agile practices, and engaging peers in honest retrospection Agile can do more to educated developers of the highest ethical conduct.

10. References

[1] Maner, W., "Unique Ethical Problems in Information Technology", *Science and Engineering Ethics*, 2:2, April 1996, pp. 137-54.

[2] Maeda, J., *The Laws of Simplicity*, The MIT Press, Cambridge, 2006.

[3] InfoWorld, "Software developer growth slows in North America", [online] [cited June 6, 2008] http://www.infoworld.com/article/07/03/13/HNslowsoftdev_1.html

[4] U.S. Department of State's Bureau of International Information Programs, "USA Economy in Brief", [online] [cited June 6, 2008] http://usinfo.state.gov/products/pubs/economy-in-brief/page3.html

[5] Joy, B., "Why the future doesn't need us", *Wired*, June 2006.

[6] "Los Angeles man admits to infecting 250,000 computers", *Reuters,* November 9, 2007.

[7] Anonymous, *Identify Theft Victim Complaint Data January 1 - December 31, 2006,* Federal Trade Commission, Washington D.C., Feb. 2007.

[8] SANS, "SANS Top-20 2007 Security Risks (2007 Annual Update)", [online] [cited November 11, 2007], http://www.sans.org/top20/#c1.

[9] NIST, "The Economic Impacts of Inadequate Infrastructure for Software Testing", [online] [cited November 11, 2007], http://www.nist.gov/public_affairs/releases/n02-10.htm

[10] Gartner, "Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO2 Emissions", [online] [cited November 11, 2007], http://www.gartner.com/it/page.jsp?id=503867

[11] Highsmith, J., *Adaptive Software Development*, Dorset House Publishing, New York, 1999.

[12] Cockburn A., *Crystal Clear: A Human-Powered Methodology for Small Teams*, Addison-Wesley, Boston, 2005.

[13] Beck, K., *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Boston, 2005.

[14] Poppendieck, M., and T. Poppendieck, *Lean Software Development: An Agile Toolkit*, Addison-Wesley, Boston, 2003.

[15] Schwaber, K., and M. Beedle, *Agile Software Development with Scrum*, Prentice Hall, Upper Saddle River, NJ, 2002.

[16] Beck, K., Beedle, M., et, al.,. Principles Behind the Agile Manifesto. [Online] [Cited: June 6, 2007] http://agilemanifesto.org/principles.html.

[17] Highsmith, J., History: The Agile Manifesto. [Online] [Cited: June 6, 2007] http://agilemanifesto.org/history.html.

[18] Beck, K., Beedle, M., et, al.,. Principles Behind the Agile Manifesto. [Online] [Cited: June 6, 2007] http://agilemanifesto.org/principles.html.

[19] McBreen, P., *Software Craftsmanship*, Addison-Wesley, Boston, 2002.

[20] ibid.

[21] Petroski, H., *To Engineer is Human,* Vintage, New York, 1992, pg 62.

[22] Florman, S., *The Existential Pleasures of Engineering*, St. Martins Press, New York, 1994, pg 6.

[23] McConnell, S., *Professional Software Development: Shorter Schedules, Better Projects, Superior Products, Enhanced Careers*, Addison-Wesley, Boston, 2004.

[24] Software Engineering Code of Ethics and Professional Conduct, Institute of Electrical and Electronic Engineers, Inc and The Association of Computing Machinery, 1999.

[25] Martin, M. and Schinzinger, R., *Ethics in Engineering*, McGraw Hill, Boston, 2005, pg 8.

[26] Baura, G. *Engineering Ethics: An Industrial Perspective*, Elsevier Academic Press, Burlington VT, 2006.

[27] Software Engineering Code of Ethics and Professional Conduct, Institute of Electrical and Electronic Engineers, Inc. and The Association of Computing Machinery, 1999.

[28] Fairweather, N., "No PAPA: Why Incomplete Codes of Ethics are Worse than None at All", *Ethics in the Age of Information Technology,* Linkoping University Press, 2000.

[29] Ambler, S., *Usable UIs*, Dr. Dobbs Journal, February 1, 2005.

[30] Software Engineering Code of Ethics and Professional Conduct, Institute of Electrical and Electronic Engineers, Inc and The Association of Computing Machinery, 1999.