Software Engineering Professionalism

WH Morkel Theunissen

Abstract— **The state of contemporary software and the practice of its development continue to raise the need for evaluating the concept of professionalism in software development. This paper investigates the definition and the concept of professionalism and in turn the resulting profession of software engineering; leading to some philosophical discussion of the subject. The elements of values, principles, practices and ethics are briefly explored. Culminating into some vision of the path forward.**

I. Introduction

As the dawn of the *Age of Connectedness* expands over the horizon and becomes our way of life today, it seems appropriate to briefly reflect on the path that lead human kind to this point and through this reflection propose some attitude changes which may assist in building an ameliorable new age.

The *Information Age* saw the mass storage, processing, distribution and retrieval of data and information. Computers were build to aid humans in accomplishing this way of life. As with previous ages of human kind each one brought the need for new skills and disciplines to light. The previous century saw the emergence of the *electrical engineer* and *computer scientist*, culminating into the *information technologist* who eventually metamorphosed into the *information communication technologist*.

As the once separate fields of communication and information technology intersected into synergy, the birth of the *Connected Age* was assured. We are at a stage where our surroundings are replete with various paraphernalia affecting our lives in minute as well as extreme ways. The majority of these articles contain some measure of software. Such software has been weaved into every fabric of life and has therefore become inextirpable.

Mastering the development and maintenance of software should no longer be considered as desirable but as essential to prevent unduly harm to human kind. The mastery of software development and maintenance falls primarily on the shoulders of the Software Engineer. As such, should we not require professionalism from this body of individuals? During a medical emergency, one is inclined to only entrust one's life to a certified medical doctor. Should one not expect the same trustworthiness from the ones who contribute to the development of almost everything we are exposed to daily? Your refrigerator, cellphone, automobile, bank account, etc.

The ensuing sections will attempt to address various components of software engineering professionalism.

II. What is Professionalism?

To understand what professionalism is, one should first gain a common understanding of the term. The natural starting point being the definition given by an English dictionary:

> "**professionalism** *n* **1** the methods, character, status, etc., of a professional. **2** the pursuite of an activity for gain or livelihood."[Collins 2000]

The above definition leads into the question of how a *professional* may be described:

> "**professional** *adj* **1** of, relating to, suitable for, or engaged in as a profession. **2** engaging in an activity for gain or as a means of livelihood. **3** extremely competent in a job, etc. **4** undertaken or performed for gain or by people who are paid. ♦ *n* **5** a person who belongs to or engages in one of the professions. **6** a person who engages for his livelihood in some activity also pursued by amateurs. **7** a person who engages in an activity with great competence. **8** an expert player of a game who gives instruction, esp. to members of a club by whom he is hired."[Collins 2000]

One would hope that the current discussion primarily refers to definitions 1, 3, 5 and 7 which one may combine into: *a person who engages in a profession with great competence*. This definition aligns with the motivation set out in the SWEBOK Guide [IEEE Computer Societ2004], which states "For software engineering to be fully known as a legitimate engineering discipline and a recognised profession, consensus on a core body of knowledge is imperative". A profession is defined as:

> "**profession** *n* **1** an occupation requiring special training in the liberal arts or sciences, esp. one of the three learned professions, law, theology, or medicine. **2** the body of people in such an occupation. **3** the act of professing; ovowal; declaration. **4a** Also called: **profession of faith.** a declaration of faith in a religion, esp. as made on entering the Church of that religion or an order belonging to it. **4b** the faith or the religion that is the subject of a declaration."[Collins 2000]

which leads to the more tangible specification provided by [IEEE Computer Societ2004]:

> "... an engineering profession is characterized by several components:
>
> - An initial *professional education* in a curriculum validated by society through accreditation
> - Registration of fitness to practice via voluntary *certification* or mandatory *licensing*
> - Specialized *skill development* and *continuing professional education*
> - Communal support via a *professional society*
> - A commitment to norms of conduct often prescribed in a *code of ethics*
>
> "

One may naively state that the first four points are primarily an infrastructure and process problem with an associated solution. However, for practical purposes one may state that the crux of professionalism for the individual lays within the last point —“A commitment to norms of conduct often prescribed in a *code of ethics*”. This brings us back to the basics of humanity, the values that are embedded in the fibre of the individual which might be shared to some degree by a group/community. Values and ethics will be further addressed in Section VI.

Another observation from the above specification is the grounding in the three pillars of engineering: technical; ethical and legal [Hoffman et al 2001]. As an engineering profession, software engineering should be build on these three pillars.

The aforementioned paragraphs provided some sound theoretical understanding of professionalism, however it may be worthwhile to reflect on the world’s notion of software engineering.

III. The World’s View on Software Engineers

The following subsections provide some interpretations on how non-software engineers perceives software engineers and computers in general.

A. *Colin Myers Interpretation*

[Myers 1995, Chapter 1] describe people’s perception on firstly computers and secondly information technologists. Myers classify people’s notion of computers as either “friend and ally” (an aid towards improving the lives of humans) or “an omnipotent threat” (where computers will take over the world, as symbolised in popular films such as “Terminator” and “2001: A Space Odyssey” and literature such as “I, Robot” by Isaac Asimov). On a more realistic level people in the past and to some extent still do see computers as a threat to their livelihood as computers replace certain functions traditionally performed by humans.

The positive or negative notion towards computers inadvertently affect people’s view on software engineers and/or information technologists. Myers categorise these views into:

The Mr Spock Syndrome: People who work with computers are seen as highly intelligent and as such unable to talk to other ‘mere mortals’. These people are seen as unable to have a good time and thus ‘boring’.

The Helpdesk Syndrome: “The false assumption is that if you know something about computers you must know everything” [Myers 1995] encapsulates a view held by numerous users. The well known scenario were a non technical person determines that the person he is conversing with is in the IT field inevitably leads the conversation towards a request for advice on hardware purchasing or fixing a desktop problem or something similar. This scenario however is not unique to the software engineering profession, it occurs in other professions such as medicine: a neural surgeon being asked a question about some hart condition. The trouble with this situation is that when appropriate answers can not be provided it leads to, as Myers put it : “Ignorance of one aspect is unfairly extrapolated to ignorance about every aspect” [Myers 1995]

The Anorak Syndrome: Software engineers are seen as arrogant and unwilling to entertain valuable suggestions provided by users and/or clients. This may be as [Myers 1995] puts it, “symptomatic of the day-to-day problems of dealing with clients, who have enough knowledge to interfere.”.

B. *Personal Interpretation*

An additional view that should be added to the aforementioned list may be **Untrustworthy**. Due to the inherent difficulties and immaturity of the discipline, combined with the numerous amateurs and amateurish behaviour of even the professionals in the discipline, the mistrust between users and software developers/providers have, rightfully so, grown. Numerous excuses may be raised by practitioners for the vast amount of flaws rampant in the software products they let loose on users. Be it business/market pressure or any other semi-valid reason; the damage to the reputation of the software engineering profession have been made and still continues to be made.

IV. How do software engineers want to be seen?

Having briefly considered the outsiders view on software engineering it may seem appropriate to raise the question of how software engineers want to be perceived? The answer to this question may vary as much as the number of software engineers, however some commonality might be found and summarised as: *to achieve the goal of engineering software that is of high quality and perceived as usable and valuable to their users, in a professional manner.*

V. How does software engineers see themselves?

Using the bold statement from the previous section, software engineers should determine if they see themselves as they want to be seen. Introspection is a valuable tool of a professional in order to grow not only as an individual but also the discipline in general. A large portion of practitioners probably feel frustrated in not being able to see themselves fulfilling the utopian role-model they defined in their dreams. Furthermore, they probably feel emasculated by the pressure placed on them by business/market forces and amateurs, both of whom do not fully support the ideals of the profession. This frustration will continue until governments and users have recognises the profession and fully support the establishment of the infrastructure (legislation etc.) required for the software engineering profession.

Software engineers themselves will need to have the *courage* to abide by stipulations set by the software engineering profession and as such face the consequences of nonconformance.

VI. Values and Ethics

One may argue that by looking at the bigger picture and the whole notion of *software engineering professionalism*, it actually boils down to the basic elements of the need for individuals and their associated common groups to conduct themselves according to a common base of *ethics* which is founded in a specific set of *values*. This section will briefly look at both these elements.

A. Values

"Values are the characteristics/qualities of something that the valuer regards as desirable and important. Humans sometimes encapsulate these into a mission and/or vision statement. Values are adhered to either consciously or unconsciously." [Theunissen 2003]

Due to the intangibility of a value, it is not easy to determine if someone has a specific value instilled in herself. Therefore some find it easier to describe values through the set of principles by which they conduct themselves. The relationship between values and principles are explored further in Section VIII.

A contemporary example of the usage of the concept of values in software development is found in the Extreme Programming (XP) methodology (see [Beck 1999; Beck 2000; Beck et al 2005]). [Beck et al 2005] defines the values underlying XP as: communication; simplicity; feedback; courage and respect [1]. These values in turn are reflected in the XP principles. Numerous other methodologies are also defined through their principles. However the principles for methodologies must be aligned to the personal principles and in turn values of their practitioners in order for them to be accepted and successfully applied.

The difficult question then becomes, what values should a professional software engineer embrace? The author propose some values, in no particular order, in the following paragraphs . Some of which have been borrowed from XP.

Continuous Improvement: A software engineer should have the urge to improve herself, her profession, the lives of users and in a broader sense the world.

Simplicity: The importance of seeking simplicity is highlighted by the following two quotes:

> "Simplicity is the ultimate sophistication." – Leonardo da Vinci
>
> "Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away." – Antoine de Saint-Exupery

and embodied in *Occam's razor*[2] which is sometimes paraphrased as "All other things being equal, the simplest solution is the best."

Quality: A professional is seen as someone who is highly competent. Thus valuing and striving for quality in all aspects of software development should be regarded as essential.

User Satisfaction: As software tend to be written for some kind of user, a software engineer should want her software to be oriented towards satisfying the need of the user.

Communication: Information need to be externalized either through the expression of intent when engineering software or when providing feedback during the execution of a program. All of these should be governed by the ideal of efficiently maximising communication between parties.

Respect: A software engineer's actions should be guided by respect on various levels: respect for oneself, the profession, colleagues, clients, customers and the environment.

Courage: Professionalism usually entails making and following the more difficult route when faced with a decision. As such a professional has to embrace the need for courage when conducting herself.

B. Ethics

The 1990's saw the establishment of a *Code of Ethics* for software engineering by the primary professional bodies of the discipline, the IEEE and ACM (see [Gotterbarn et al 1997; ACM/IEEE-CS]). Establishing this code was an important step on the road towards the recognition of the software engineering profession. Analysis of this code suggests that it reflects the values as defined in the previous subsection. In summary it addresses the conduct of the professional towards: the public; the client and employer; product; management; profession; colleagues; oneself and finally provide guidance on the professionals judgement.

VII. Practicality of being Professional

The previous sections have touched on the theory of software engineering professionalism, however on the path forward the practicality thereof will be the biggest challenge facing practitioners. Beyond the provision of (1) a professional society and (2) the initial and continued educational system, both of which has been established to some extent, a legal infrastructure is required.

Legislation of the profession will bring *liability* and *enforcement* to the playing field. In practice it will require the full buy in from all the stakeholders involved in software engineering. Recognised software engineers will need to realise and accept the 'burden' this will place on them.

Liability: One such burden will be the acknowledgement that one will be liable for one's actions and the product being delivered. The accepted notion by software developers and users that *bugs* in software is the norm rather than the exception needs to be corrected. Bugs should be seen by all stakeholders for what they are, unacceptable flaws. A collapsing bridge is not shrugged of by stakeholders as a natural occurrence of bridge building, it instead leads to serious liability investigations and subsequent consequences.

Enforcement: Liability goes hand-in-hand with enforcement. Whether it is the revoking of the license of a licensed practitioners or taking legal action against clients who did not use licensed practitioners for a project of the nature requiring professionalism.

VIII. The Foundation

There exists an intricate relationship between values, principles, practices and ethics. This inter-relationship need to be in harmony for the effective realisation of professionalism. However this realisation could take on various forms as each unique combination of external forces to the software development effort will require an optimal balance of these elements. Software engineering rarely takes place in a vacuum, but instead is driven by other disciplines such as aeronautics, military, medicine or accounting to name a few. These differnces should be incorporated into the different realisations.

[1] The reader is referred to [Beck et al 2005] for a detail discussion on each of these.

[2] William of Occam (or Ockham) (1284-1347) stated "Entities should not be multiplied unnecessarily."

IX. To be or not to be?

"To be, or not to be, that is the question" – William Shakespeare's Hamlet, Prince of Denmark, Act III, Scene I.

Just as Hamlet was confronted with the choice between action or no action, software developers are faced with the question of embracing professionalism or continuing with the *status quo*. Accepting and striving for professionalism will not be without cost and struggle, but the hope should be that it is for the better of society and oneself.

Some lingering questions remaining are: what will the effect be of adopting professionalism? Will it stifle innovation? Are we going to see a reduction in the speed of delivery of new and exciting products? What about amateurs?

Considering that we are entering the *Age of Connectedness* we need to ensure that the seeking of professionalism remains in line with the reality of thousands of applications being developed at internet speed/time for a diverse range of needs. This diversity will hopefully ensure the coexistence between both software development amateurs and professionals in some symbiotic form, feeding and driving one another.

References

[ACM/IEEE-CS]
ACM/IEEE-CS, 'Software Engineering Code of Ethics and Professional Practice (Version 5.2)', Online, Online: http://www.acm.org/about/se-code (accessed: 2008/01/18), ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices(SEEPP).

[Beck 1999]
Beck K, 'Embracing change with extreme programming', *IEEE Computer*, vol. 32, no. 10, pp. 70–77, October 1999, ISSN 0018-9162.

[Beck 2000]
Beck K, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2000, ISBN 201-61641-6.

[Beck et al 2005]
Beck K and Andres C, *Extreme Programming Explained: Embrace Change*, The XP Series. Addison Wesley, 2nd edn., 2005, ISBN 0-321-27865-8.

[Collins 2000]
Collins English dictionary: 21st century edition, HarperCollins Publishers, 4 edn., 2000, ISBN 0 00 472529-8.

[Gotterbarn et al 1997]
Gotterbarn D, Miller K and Rogerson S, 'Software engineering code of ethics', *Communications of the ACM*, vol. 40, no. 11, pp. 110–118, 1997, ISSN 0001-0782.

[Hoffman et al 2001]
Hoffman DM and Weiss DM (editors), *Software Fundamentals: Collected Papers by David L. Parnas*, Addison-Wesley, 2001, ISBN 0-201-70369-6.

[IEEE Computer Societ2004]
IEEE Computer Society, 'SWEBOK: Guide to the Software Engineering Body of Knowledge', Tech. rep., The Institute of Electrical and Electronics Engineering, 2004, Online: http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf (accessed: 2006/02/15).

[Myers 1995]
Myers C (editor), *Professional Awareness in Software Engineering*, McGraw-Hill, 1995, ISBN 0-07-707837-3.

[Theunissen 2003]
Theunissen W, *A case-study based assessment of Agile software development*, Master's thesis, University of Pretoria, 2003, Online: http://upetd.up.ac.za/thesis/available/etd-07152004-084708/ (accessed: 2004/10/18).