

Changing Culture:

Should you? Could you? How? Why?

London, June 2009

C. C. Shelley
OXFORD SOFTWARE ENGINEERING Ltd
9 Spinners Court, 53 West End,
Witney,
Oxfordshire
OX28 1NH
www.osel.co.uk
info@osel.co.uk
Tel. +44 (0) 1993 700878

It should be borne in mind that there is nothing more difficult to handle, more doubtful of success, and more dangerous to carry through than initiating changes in a state's constitution. The innovator makes enemies of all those who prospered under the old order, and only lukewarm support is forthcoming from those that would prosper under the new. Their support is lukewarm partly from fear of their adversaries, who have the existing laws on their side, and partly because men are generally incredulous, never really trusting new things unless they have tested them by experience. In consequence, whenever those who oppose the changes can do so they attack vigorously, and the defence made by the others only lukewarm. So both the innovator and his friends come to grief.

– *Machiavelli: The Prince, Part VI*

Contents

- **Development of s/w cultures**
- **Mapping a s/w culture**
- **Why change?**
- **How to change - tools**

Software Cultures

- **Software development is now more than fifty years old**
- **Software cultures have developed over that time:**
 1. **Initially engineering / scientific – (elite, ‘white coat’)**
 2. **Split into ‘engineering’ (computer makers and embedded systems), and ‘DP’ - now known as ‘IT’ (predominantly computer users – mostly mainframe in commercial environments - part of the organizational infrastructure)**
 - **also a hacker sub (deliberately and consciously ‘sub’) culture, exemplified by the Unix, and now Linux and OS worlds**
 3. **Availability of mini and micro computers, and later, the internet dilutes, or marginalizes, engineering culture and disseminates a wider ‘de-professionalized’ culture –**
 - **(some striking analogies to earlier ‘telegraphers’ culture in second half of 19th century)**

Describing Cultures...

- **It is proposed that an understanding and ability to communicate a culture is a prerequisite to any attempt to change it.**

- **A useful place to start is the scope...**
 - **Which culture(s) are we discussing?**
 - **the host organization**
 - **the software / IT department (in the organization)**
 - **Management culture**
 - **Team culture**
 - **analysts or developers or testers (in the department)?**

 - **What are the relationships between these?**

 - **How should they be aligned?**

...Describing Cultures...

- **Alignment...**
 - **What organizational culture would you expect to find in a:**
 - **Japanese computer manufacturer**
 - **American bank**
 - **Dutch insurance company**
 - **British defence contractor**
 - **Scandinavian pharmaceutical company**
 - **French telecoms company.....**

 - **What is the most appropriate ('best') software culture for these orgs?**
 - **w.r.t. to host organization culture?**
 - **the host orgs requirements of the software and software department?**
 - **both?**
 - **other?**

...Describing Cultures

- **...Alignment...**
 - **Software development team/department needs:**
 - **Primarily: a software culture that enables a competence to adapt and change to deliver optimal solutions to the host organization (note: this is not meant to imply 'agile')**
 - **Secondary: alignment to the business, as a result of the first**

(‘Avoiding the Alignment Trap’: MIT Sloan Management Review)

- **(An aside: it is useful to evaluate how software staff view themselves:**
 - **as part of <org,> who happen to be in software**
 - **or as software professionals, who happen to work for <org.>)**

(which is best?)

A method for mapping* a software culture...

- If a culture is to be managed or changed it is essential to be able to characterize it 'as is' and/or as it should be 'to be'
- How?
 - Often described as a scale or matrix
 - Constantine – closed/open, synchronous/random
 - DeGrace, Stahl - Greek/Roman
 - Others

these are interesting, but constrained by the framework

- Or assumes 'good'
 - Wiegiers – 'Creating a Software Engineering Culture'
 - McCarthys – 'Software for your Head'
 - Kerth - 'Project Retrospectives'

...A method for mapping a software culture...

- **To measure a culture ('as is, or 'to be', or 'compared with')....**
- **Simple three stage process (to be applied discreetly)**
 - **Stage 1: Build 'unconstrained' list of attributes**
 - **Stage 2: Check attributes**
 - **(Optional) Stage 3: Compare and analyse checked attributes**

...A method for mapping a software culture...

- **Stage 1:...**
 - **For the culture of interest build list of relevant attributes**
 - **Where 'relevant' is:**
 - present,
 - absent but wanted,
 - seen elsewhere
 - desired,
 - to be avoided
 - **Build list**
 - **Use 'seeding' list** (see example)
 - **borrow attributes from other models**
 - **Organize list if desired** (see example attribute sheet)
 - **Values**
 - Attitudes
 - » Behaviours

...A method for mapping a software culture...

- **...Stage 1:**
 - **Now build list for culture's context (go up a level)**
 - **e.g. if first list is for software department then repeat for organization as a whole**

 - **Now build list for 'producers' within the culture (go down a level)**
 - **E.g. if first list was for projects then repeat for individuals within projects**

 - **Now have three lists of varying granularity with culture of interest 'sandwiched'**

(It may be of interest simply to compare the selected attributes of the three cultural 'layers' before stage 2 – how do they align? anything interesting?)

...A method for mapping a software culture...

- **Stage 2:...**
 - **For each list of the three lists check each attribute as:**
 - **'present'**
 - **'mixed'**
 - **'not present'**
 - **- and comment, if desired; recommended – builds a richer picture**
 - in the organization 'as is' to establish a cultural baseline**
 - **Allow several individuals to check the lists and reconcile with Delphi like process**
 - **Modify attributes and add new ones as needed or discovered**

...A method for mapping a software culture

- **...Stage 2:**
 - Repeat for 'to be' culture - or 'as they are', or 'as we were'...
 - **Outcome:**
 - **A record and basis for comparison of the culture, and its context and producers**
 - **record has a unique structure**
 - so is 'value free' - no presumption of 'better'
 - and no score
 - so no target syndrome (like CMMI)
- **Stage 3:**
 - **Analyse, and communicate, and compare culture**
 - With context - for (mis)alignment
 - With producer - ditto
 - With comparable ('like them', 'like we were') or exemplar cultures

Why change a software culture?...

- **Software cultures emerge unconsciously**
- **'an organization gets the software it deserves'**
- **software = team**
- **change is risky**
- **change is uncomfortable for most people**
- **change will disrupt and displace accepted norms and values**
- **it is not clear what 'better' means**

...Why change a software culture?...

- **There must be a compelling reason to attempt cultural change:**
 - **The current culture is inhibiting or damaging to software development and it is necessary to replace or realign it *so that software developers and others are better able to produce and maintain software systems that, measurably, better meet the organization's needs*, whether the software is for use by the host organization or is a product of the host organization.**
 - **(and it is cost effective and feasible to change the culture)**

...Why change a software culture?

- **Good advice:**
 - **Don't**

How to change a software culture...

- Ignoring that good advice, the following tools for change have been used:
- They are:
 - best described as patterns:
 - Fashionable (cf methods, procedures, processes)
 - Describe solutions to problems (not just ad hoc ‘improvement’)
 - Describe these solutions “...in such a way that you can use the solution a million times over without ever doing it the same way twice.”
 - (Alexander, 1977)
 - classified:
 - P (pattern) or A (antipattern) or P/A (both – a double edged tool)
 - U (develop understanding of culture) or C (make change)
 - tend to be rather indirect, or very direct, even brutal

...How to change a software culture...

1. Mapping tool described earlier (P, U)
2. Senior Management (P, C)
3. Remove leaders and influencers (P, C)
4. Disband teams (P, C)
5. Reorganization (P, C)
6. CMM & CBA IPI (P, U)
7. CMM common features (P, C)
8. Mimicry (P/A, C)
9. Education and training (P, C&U)
10. Skeptics (P, U&C)
11. Kotter's eight stage change process (P, C)
12. Performance targets (A/p, C)
13. Rapid Process Improvement (RPI) (P, C)
14. Hawthorne effect (P, C)

...How to change a software culture...

- **2. Senior Management (P, C)**
 - **and opinion leaders, are one of the most effective agents of change**
 - **but often don't know it**
 - **use them to direct change**
 - **they often don't know what to do – help them**

...How to change a software culture...

- **3. Remove leaders and influencers (P, C)**
 - **...that work against desired culture**
 - **and replace with those that work towards it.**
 - **make the change conspicuous, and the reason for it clear**

...How to change a software culture...

- **4. Disband teams (P, C)**
 - **...that work against the desired culture**
 - **and rebuild new teams**
 - **visibly and with the reason known**

...How to change a software culture...

- **5. Reorganization (P, C)**
 - **Structure implies values**
 - **Can introduce uncertainty and chaos that can provide a background to crystallize out the new culture**

...How to change a software culture...

- **6. CMM & CBA IPI (P, U)**
 - **CMM provides a framework for understanding organization ‘types’**
 - **CBA IPI is a good diagnostic tool – and can provide impetus for change**

...How to change a software culture...

- **7. CMM Common Features (P, C)**
 - **Is a set of activities and organizational infrastructure (commitment, abilities, measurement and analysis, and verification) that provides a way of ‘institutionalizing’ or ‘bedding in’ desired behaviours**

...How to change a software culture...

- **8. Mimicry (P/A, C)...**
 - **a common learning technique, requiring repetition or practice**
 - **Can be useful for (mostly) technical changes...**

“The bad news is that, in our opinion, we will never find the philosopher’s stone. We will never find a process that allows us to design software in a perfectly rational way. The good news is that we can fake it. ... ”

– *David Parnas, ‘A Rational Design Process: How and Why to Fake It’.*

...How to change a software culture...

- **8...Mimicry (P/A, C)**

- **But usually goes wrong for complex or major changes, e.g. CMMI, agile**
- **Sterile and damaging imitation...**

... In the South Seas there is a cargo cult of people. During the war they saw airplanes with lots of good materials, and they want the same thing to happen now. So they've arranged to make things like runways, to put fires along the sides of the runways, to make a wooden hut for a man to sit in, with two wooden pieces on his head for headphones and bars of bamboo sticking out like antennas— he's the controller— and they wait for the airplanes to land. They're doing everything right. The form is perfect. It looks exactly the way it looked before. But it doesn't work. No airplanes land. So I call these things cargo cult science, because they follow all the apparent precepts and forms of scientific investigation, but they're missing something essential, because the planes don't land.

– *Surely You're Joking Mr. Feynman?*, *Richard Feynman*

...How to change a software culture...

- **9. Education and Training (P, C&U)**
 - **As a diagnostic tool**
 - **An opportunity for dialogue between trainer and trained***
 - **As a catalyst for change**
 - **A signal that <subject of training> is now expected/permitted**
 - **(And occasionally as a means for transferring knowledge and skills, and to influence values, of course)**

...How to change a software culture...

- **10. Skeptics (P, U&C)**
 - **Oppose the change**
 - **Can be smart, influential, experienced....**
 - **Seek them out (they may need to be hunted down and cornered)**
 - **Understand their beliefs and opinions (they may be right)**
 - **Work to consensus, or...**

...How to change a software culture...

- 11. Kotter's eight stage change process (P, C)
- In particular step one 'establishing a sense or urgency' (or precipitating a crisis) is often important if the desire to change is to be taken seriously.
 - Without it day to day operational matters dominate people's thinking.
- The remaining steps: *create a guiding coalition, developing a vision and strategy, communication the change vision, empowering broad-based action, generating short term wins, consolidating gains and producing more change, anchoring new approaches into the culture* are about developing sharing and sustaining the changes.

...How to change a software culture...

- **12. Performance Targets (A/p, C)**
 - **These usually cause damage.**
 - **Poorly thought out performance targets will trigger dysfunctional behaviours, undermine development activity and hide real performance.**
 - **Typical performance measures that trigger dysfunctional behaviour:**
 - the achievement of, say, CMMI ML3 by a given date,
 - year on year productivity improvements,
 - or both together
 - **(Several instances of both these targets being inflicted on software developers a have been observed by the author.)**
 - **HP's '10X' performance targets for its software development (requiring a carefully defined ten fold improvement in quality over five years) was a notably successful programme of improvement driven by performance targets.**

...How to change a software culture...

- **13. Rapid Process Improvement (RPI) (P, C)**
 - An approach, supported by a set of tools for making change in organizations. Most of these changes are technical in nature. RPI is a form of SPI which has a high technical content.
 - RPI characteristics: it is focussed on problem solving, not unspecified 'improvement', desired results are specified and measurable, it is driven by results and learning from them, which are delivered by ongoing small increments of work, work is timeboxed, work may use models or frameworks but not driven by them, work is inclusive and integrated into day to day work, work is structured by the use of methods and tools, it supports local ownership and accountability, and is opportunistic.
 - Tools include methods for exploring and defining processes, problem solving, fixing process problems, performing post implementation reviews (the tools have been developed and acquired and are not necessarily original – just useful), defining measures, etc. Many are procedural which makes them potentially easy to understand, but appear to be unfashionable.
 - www.osel.co.uk/rpi/rpi.htm

...How to change a software culture...

- **14. Hawthorne Effect (P, C)**
 - ... is usually considered as an oddity and a confounding factor when trying to understand how to improve ways of working. Consider using the Hawthorne effect as a tool for change in its own right.
 - Close observation and genuine interest in ways of working, with a real concern with improvement and perhaps nominal changes or placebos can trigger disproportionate improvement, steered by observation of the results and careful selection of the 'treatments'.

...How to change a software culture

- (Cultural mixing
 - Take two cultures and combine
 - Forms an ‘alloy’ with attributes not present in either contribution cultures
 - e.g. academic/military – odd, occasionally comical - but formidable)

Envoi:

“ How easily men could make things much better than they are, if they only tried together.”

Churchill to Clementine - 1909

**OXFORD
SOFTWARE ENGINEERING
LIMITED**

9 Spinners Court, 53 West End,
Witney,
Oxfordshire
OX28 1NH

www.osel.co.uk
info@osel.co.uk
Tel. +44 (0) 1993 700878

... But to discuss this subject thoroughly we must distinguish between innovators who stand alone and those who depend on others, that is between those who to achieve their purposes can force the issue and those who must use persuasion. In the second case they always come to grief, having achieved nothing; when, however, they depend on their own resources, then they are seldom endangered.

— *Machiavelli: The Prince Part VI*