Value Engineering – the long and the short of it

Alan Dix
Computational Foundry, Swansea University, Wales

https://alanadix.com/academic/papers/value-engineering-2020/

Abstract: You can brow-beat people into doing what you want, try to persuade them to go along with you, or subtly fool them. However, it is usually far better to accept that people have different drives, aspirations and motivations and then design systems and process that achieve organisational goals given the particular personal values that drive individual decisions. This is value engineering. Through a series of short case studies, this article outlines certain aspects of human behaviour that I have found valuable in both theoretical understanding and practical application. We will see that the basic principle of understanding people's values can be applied at multiple levels from large-scale organisational processes to individual moment-by-moment decisions. We will also see that it is important to understand human values at different timescales, a fact noted by Rousseau in the 18th century and still pertinent in designing for climate change or pandemic today.

Keywords: human–computer interaction, individual values, organisational design, user experience

Introduction

This article is a personal account, not of a particular piece of cutting-edge behavioural science, but rather of certain aspects human behaviour I have found useful over a 35-year career in human–computer interaction research and practice. Centrally it concerns individual values, not in the ethical sense, but the particular motivations, drives and goals that determine our moment-to-moment decisions and actions.

Applying this in practice is a two-step process.

1. The first is to understand the individual values that drive decisions and behaviour at different scales and times, recognising that each individual is different.
2. Second, once we understand them, is to design system structures and processes that achieve organisational goals given these individual values. That is *value engineering*.

The term 'value engineering' initially sounds insidious bringing to mind advertisers manipulating our purchasing decisions, or paternalistic government behavioural science units, inspired by Thaler and Sunstein's "Nudge" [TS08], influencing our choices in health, finance and maybe even politics. However, it can be transparent and something we can do to ourselves, for example, using smaller dinner plates to reduce portion size.

We can see a real example of this working in Box 1. The system analyst spent time observing machine operators, realised that keeping their clothes clean was a major priority, and then made a small intervention that achieved the organisational goal of higher throughput, working with, not against, the operators' own values [Anon].

There are many different people involved in the software production process: clients who procure a system, potential stakeholders interviewed/consulted as part of design, within the software company: developers, designers, analysts, marketing and management; after production the primary users, documentation they use, help centres and maintenance. We may want to apply value engineering, or indeed the other behaviour science insights in this Special Issue, to the actors at different stages of this process.

The interventions we can consider range in scale from a whole organisation to individual actions, and also over different time scales. We'll start by looking at the largest scale and the work down, each level illustrated by a story of practical application. However, every large-scale intervention depends on numerous personal choices, so this is partially about the lens with which we examine activity.

Box 1: The printer and the ink

The first 'proper' computer book I ever read was a 1960s book on Systems Analysis [Anon] that I found in a public library when on a seaside holiday one summer. I have no idea of the author or title, but it profoundly influenced my own notions of computing.

The author told of an assignment to a company where they had recently had delivery of a new printing machine. The machine was not achieving the desired throughput and they were wondering whether they needed to upgrade further to a computer-controlled one. After a short period he went back to the management and said, "For a budget of 100 dollars a year I can solve your problems". His clients were amazed as they had been expecting to spend tens of thousands on a new computerised printer, but they followed his advice and the throughput increased markedly.

What did he do?

Before thinking about a solution, the author had spent some time observing the printer and those using it. In particular the direct machine operators were all young women, and he saw that they approached the machine very tentatively – it was a printer with lots of ink and they were being careful of their clothes. The hundred dollars bought white overalls for them to wear. Once their clothes were protected, they worked more freely, and the machine operated to peak efficiency.

Large scale – organisational structures

In the UK through the 1950s and 1960s, it was common for employees to spend their whole working life with a single company; maybe starting as an apprentice and then gradually working through the ranks, with on-the-job training and day release schemes. There was often a sense of loyalty to the company, and a sense of working to a common goal, aided no doubt, by this being a time of diminishing inequality across society and post-war optimism. In Asian factories, this was often even more apparent with morning corporate exercise sessions – beloved by documentary filmmakers of the time.

This is a form of *value alignment* – workers and bosses with a shared goal.

As we shall see later in this article, these shared high-level goals may not always lead to mutually beneficial behaviour, but are often a touchstone for success. Certainly, over the years whenever I've scrutinised a contract, I look less for complex protective measures and more for those that create the grounds for win-win situations: where the other party's gains mean that my university or organisation gains, and vice versa. If this is the case, then there may still be some disparity, but in general if the other party follows their best interest, your own institution gains.

In such contracts it is not that you actually share the same underlying values, but that they create a structure within which you can each pursue your individual goals and as a side effect benefit the other.

In general we can engineer such situations, even when dealing with large numbers of individuals with disparate values. Rather than convincing employees, end-users or customers to adopt our own values, we accept the differences and work with them. This is just the same as when you work with any physical material; you engineer outcomes based on what is there, not what you'd like to it be!

In the example in Box 2, when faced with a complex situation (ten group heads with six staff and 250 modules), a market-like system allowed group heads to allocate their own staff, satisfying their own values for autonomy and equity, whilst still satisfying the university's organisational goals of efficient teaching. That is we achieved alignment of outcomes without identical values.

As well as applying such principles to the systems we create, we can also apply them within the software engineering process. For example, at one point it was reported that Microsoft development teams were responsible for the maintenance costs of the software they produced. There is nothing like knowing you are going to be responsible for bug fixing to ensure well-engineered code! I don't know how long this practice persisted nor how successful it was; as we'll see later, long-term value alignment does not necessarily lead to short-term behaviour. Maybe though, this is something that should be applied to CEOs making their bonuses linked to 10-year results, not 3-5 year periods of office.

Box 2: A market for timetabling

Some years ago I had a senior position in a university computer department. The department had previously been three separate divisions of 20 staff, each of which was responsible for around 80 teaching modules including some first year, some more advanced undergraduate and some masters teaching. Each division had managed its own assignment of staff to teaching, with an occasional exchange of staff between the divisions, and this had worked well.

Just prior to my own appointment the three divisions had been merged into a single unit with 60 staff divided into ten research groups of different sizes, which each took responsibility for a small number of modules pertinent to their area. The aim was to maximise the autonomy of the groups, but the topics of the groups meant that some had more large-class-size 'low level' modules than others and some were starting to create many new very specialised modules, each of which would only have a few students each. Our challenge was to somehow allocate 60 staff to around 250 modules and manage new module creation in ways that ensured efficient teaching whilst maintaining group autonomy.

We kept discussing this at management meetings and then putting off the decision. As the new appointee I initially assumed my colleagues knew better, but eventually offered to have a go at the problem.

I gathered the group heads together and outlined a sort of market, with two student FTE 'currencies' one for large-class modules, one for the rest. Each group head had a balance sheet, in the credit side was the student FTEs for the modules they were responsible for, on the debit was the number of student FTEs that needed to be taught by the group, taking into account their staffs' available time. Their job was to balance their books, negotiate between themselves who taught what and update a master spreadsheet.

The system was clearly transparent and fair and there were no objections. Over the coming month the ten independent group heads allocated their 60 staff to the 250 modules and only on three occasions was I called on to help when there was a difficult assignment.

It worked so well because of *value engineering*: the process allowed the group heads to pursue their own value systems of autonomy whilst as a side effect achieving the organisational goals of effective teaching.

Medium Scale – products and customers

When dealing with our own software development teams or with organisational employees, we can, to some extent, impose the systems and procedures that we believe can create beneficial outcomes.

However, when offering a product to end-users who have choice, *adoption* is a core issue.

This was recognised quite early in the CSCW (computer-supported cooperative work) literature. Jonathan Grudin wondered why so many early groupware systems failed, and identified several factors [Gr88], some very pertinent to this article. First there was often a disjunction between cost and benefits; that is a failure of value alignment discussed above. Equally important were issues of critical mass. It is clear nowadays how useful it is to have a phone, email, and instant messaging: everyone has them, so if we want to communicate with someone it is sensible to use them ourselves. But what about the very first person to buy a phone, or adopt email?

Once a *critical mass* of users have been reached, the value for each user exceeds the cost of purchase/installation/learning, leading to self-sustaining adoption, but the difficult part is designing *zero-point value*: the reason for the very first user to adopt before anyone else does so. More generally designers often focus on the long-term benefits once their products are adopted but neglect the *path to adoption* [DX08].

In the 1980s ICL promoted data dictionaries to its customers as this would enable far more efficient and reliable system creation, but the only obvious way to adopt them was en-masse, a large-scale re-engineering of one's code base; not surprisingly the local authority for which I worked at the time continued with a system of paper documentation of file structures and copy-and-paste of data definitions between COBOL programs. In the 2000s the PHP community faced exactly the same hurdle with a code-breaking change in semantics between PHP versions 4 and 5. Because the same syntax had different meaning, this could not even be detected by run-time errors, but required literally examining code line by line. Not surprisingly it took at least five years for many open-source code-bases to adopt the new version, and I was aware of commercial websites still using PHP 4 ten years after it was deprecated. In both cases the lack of an incremental conversion strategy meant there was no clear *path to adoption*, no matter how good the end point would be.

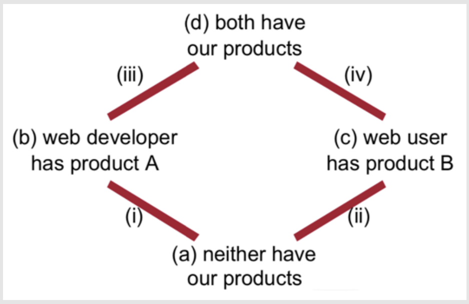
Understanding these issues allows one to deliberately engineer-in zero-point value and design paths for adoption. The earliest explicit example of this that I know of was the work of Andrew Cockburn and Harold Thimbleby [CT93] who created an email system that employed threading of email conversations that worked particularly richly when multiple users had the same system, but used automatic threading to provide zero-point value, fully 20 years before this was available in Gmail!

In dot-com years we extended this thinking to products with multiple interlinked users groups (see Box 3), analysing the *market ecology* of how each user group's adoption would influence the choices of others. We deliberately designed a *lattice of value* [DX01] for each product we created, offering zero-point entries for each user group and engineering mutual value gains for the interactions between adopters. Although we were unaware of this at the time, some of these issues had been studied by economists analysing the growth of spreadsheet use [BK96]; the theory of *network externalities* subsequently became critical in anti-trust cases against Microsoft in the early 2000s and in understanding the growth of hi-tech mega-corporations since.

Box 3: The lattice of value

In 1998 I was part of a dot-com era start-up. The word 'viral' was still a purely medical term and we needed to work out our own ways to understand the different properties of an interconnected digital marketplace. We realised that there were multiple people involved in the success of a new product, for example, in an educational context teachers, pupils and parents, each of which would influence the adoption decisions of each other. As a platform provider we were also interested in the interactions of developers using our platform and their end users.

It was comparatively easy to think of interventions that would work once a substantial number of users and developers used our platform: if there were lots of developers using it their users would follow, if there were lots of users already using it then developers would build for it. Many innovations we saw stopped there … and failed.



To address this, for each product we would create a 'lattice of value' [DX01]. We were at location (a) – no users and no developers, but we were aiming for (d) – lots of both. In order to get from (a) to (d) we either needed to traverse route (i)–(iii), or (ii)–(iv). For each product we designed, we aimed to have both paths. Steps (iii) and (iv) were the easy cases of getting users once we had developers on board and recruiting developers to an existing user base. We focused particularly on the bootstrapping steps: (i) how did we offer value to the first developer with no existing user base; and (ii) how did we offer value to the first user with no third-party developers.

We were inspired partly by the success of Adobe PDF – Adobe offered free PDF readers that could be distributed on CDs (and later the web) alongside documentation written in PDF. That is they offered a path for the first developer (i).

Small Scale – from motivation to behaviour

An increasingly important area within human–computer interaction research, and indeed social policy in general, is the design of systems to change behaviour addressing areas including physical health, mental heath and environmental action. One focus of this are materials that in various ways alter our knowledge and attitudes in relation to a topic and so increase motivation to take beneficial actions. This is not new; indeed the game Monopoly was originally designed with two sets of rules, one competitive and one cooperative, in order to instil the insidious dangers of the former! However, one of the truisms of this form of research is that there is a vast gulf between motivation and behaviour – this is evident in the issue of climate change where surveys consistently show younger people more environmentally aware, but, in fact it is the older demographic who are more likely to take actual climate-beneficial action (with the one exception of becoming vegan) [Av20].

Why this divide between motivation and behaviour?

The French philosopher Jean-Jacques Rousseau identified this conundrum in the 18th century [Ro62]. Rousseau distinguished two types of will, often called *real will* (deep, long term desires) and *actual will* (momentary drives). He gives the example of passing a jewellery shop and noticing a beautiful ring. He argues that your actual, momentary will is to smash the window, grab the ring and run; however your real, deeper will, is to live in a well ordered society where all people are free of the fear of theft. The difficulty is that decisions are made in the moment, so that actual will may act against real will. Rousseau uses this to promote the need of state intervention from libertarian principles, however this applies more widely. You may be on diet and really want to lose weight (*real will*), but surely that one cookie, won't make a difference (*actual will*). Or during Covid-19 lockdown you know that everyone should reduce travel to halt the disease and save lives (real will), but of course the trip you want to make, just this once, is obviously necessary (actual will).

Note that this distinction is different from the two types of conscious/unconscious thinking popularised in Kahneman's "Thinking Fast and Slow" [Kh11]. Unconscious reactions and unthinking habits are one of the reasons for the gulf, but at the moment you ask that question "*surely this one cookie won't matter?*" you are making a conscious decision.

The reasons for the real–actual will gulf are multiple: some is about those unconscious, fast mode actions; some is because we heavily discount the future (in the wild, tomorrow you are dead anyway, so not worth worrying!), some is because individual actions now only make a tiny and uncertain impact on the larger picture (just one plastic bag).

Crucially however, in recognising the gulf, we can engineer our systems to bridge it.

One of the simplest design maxims is that if you want somebody to do something, make it easy. In the days before GDPR websites exploited this by making personal information sharing the default. However, you can exploit this in more positive ways; one of these is *Micawber management* (named after Dickens' famous procrastinator [Di50]) or *positive procrastination* – making it easy to put off hard things until a more suitable moment. For example, during software development, at the point you are coding and need to make a choice, maybe the default colour or size of a message, you know that you should put this in a system configuration option, but in the rush of coding you just define a constant. So, provide a simple comment tag "`**CHECK**`", that can be added rapidly allowing coding to continue, but subsequently automatically detected and highlighted for review.

Another general strategy is to provide means for users to sign up in times of reflection and leisure based on their long-term motivations, which will in some way constrain or modify their later momentary decisions and behaviour. Gamification often works in this way. Box 4 describes the development of OpenBadges [MP10, OB], for Peer-to-Peer University [P2PU] a community with a very high-level of intrinsic motivation to learn (real will), but who still found it useful to add badges to provide on-going encouragement and incentive (actual will).

Within the software development process, many aspects of Agile processes also exploit the power of short-scale motivation for coder (with sprints) and clients (with frequent deployments)

Box 4: OpenBadges

P2PU (Peer2 to Peer University) allows its members to create new courses and follow each other's courses using a peer education model [P2PU]. The course provider may set the syllabus, but is not necessarily an expert in the area, merely sets and agenda for mutual learning. There is no compulsion to join any course and so those participating are all highly motivated.

Despite this high level of motivation P2PU developed a system of badges, which was subsequently adopted as the OpenBadges project by Mozilla [MP10, OB]. OpenBadges allow you to display your achievements as a web badge, rather like Guides and Scouts sew achievement badges onto their uniforms.

It is well understood within pedagogic and economic theory that internal motivation (our own goals and aspirations) is far more powerful than external motivation (externally imposed goals) [BT03]; this is one of the reasons that teachers try to make lessons fun as well as informative.

Given a highly internally motivated membership why did P2PU create what is essentially a form of external motivation? In fact, the apparent contradiction makes sense once one takes into the account the difference between Rousseau's real and actual will. While a course member might want to complete the course at a large scale (*real will*), in the moment during busy lives, they constantly gave other things priority (*actual will*). The gamification of OpenBadges offered short term motivation to help them achieve their long-term goals.

Putting It Together

Although these scales have been presented separately, they of course interact. For example, making development teams responsible for long-term maintenance; solves the problem of value alignment instilling suitable long-term motivation, but leaves open the discounting of future cost. Programmers may still cut corners to 'get things out', even though they know it is building technical debt. The "`**CHECK**`" tag or adopting Agile practices may be ways to translate the long-term desire to create robust code into momentary practices.

In short, whether it is your own day-to-day lives, the coding practices of your developers or the adoption of end-users: take time to understand the values of people, know that these operate at multiple granularities, and engineer your processes to ensure that each person's moment-to-moment value decisions work towards the desired common goal.

Take Aways

- Seek to understand the personal values of people involved in the software production process
- Recognise that long term motivations and short term behaviour are different – decisions happen in the moment
- Design structures, processes and systems so that individuals pursuing their own goals naturally lead to desired organisational outcomes.

References

[Anon] (1960s or early '70s) The amazing book about systems analysis in the public library at Weston-super-Mare. If you ever find this book, please let me know.

[Av20] Aviva (2020). "Generation woke? Over 55s most likely to recycle, study shows". 13 Feb 2020. https://www.aviva.co.uk/aviva-edit/your-things-articles/generation-woke-over-55s/

[BT03] R. Bénabou, J. Tirole. "Intrinsic and Extrinsic Motivation", The Review of Economic Studies, 70(3):489–520, 2003. doi: 10.1111/1467-937X.00253

[BK96] E. Brynjolfsson, C. Kemerer. "Network Externalities in Microcomputer Software: An Econometric Analysis of the Spreadsheet Market". Management Science, 42(12):1627-1647, December 1996.

[CT93] Andrew Cockburn , Harold Thimbleby, "Reducing user effort in collaboration support", Proc. of the international workshop on Intelligent user interfaces, p.215-218, 1993,

[Di50] C. Dickens. "David Copperfield", 1850.

[DX01] A. Dix. "The lattice of value – designing products for self-growth". eBulletin. 2000. (accessed 5/4/2020). https://www.magisoft.co.uk/alan/ebulletin/lattice-of-value/lattice-of-value.html

[DX08] A. Dix. "Designing for adoption and designing for appropriation". Talk at at University of Technology, Berlin, 12th Feb 2008. https://www.alandix.com/ academic/papers/berlin-talk-feb-2008/

[Gr88] J. Grudin. Why CSCW applications fail: problems in the design and evaluation of organizational interfaces. Proc. of the 1988 ACM conference on Computer-supported cooperative work (CSCW '88)., pp. 85–93, 1988. doi: 10.1145/62266.62273

[Kh11] Daniel Kahneman (October 25, 2011). Thinking, Fast and Slow. Macmillan

[MP10] The Mozilla Foundation and Peer 2 Peer University, in collaboration with The MacArthur Foundation. Open Badges for Lifelong Learning. 2010. (accessed 29/3/2020) https://wiki.mozilla.org/ File:OpenBadgesWorking-Paper_012312.pdf;

[OB] OpenBadges. (accessed 27/1/2020). https://openbadges.org/

[P2PU] Peer 2 Peer University. (accessed 27/1/2020). https://www.p2pu.org/

[Ro62] J-J Rousseau. "The Social Contract". 1762.

[TS08] R. Thaler, C. Sunstein. "Nudge: Improving Decisions about Health, Wealth, and Happiness". Yale University Press. 2008.