AGLPM1 - Unit 1 - ACTIVITY 2: OBSERVE

The 12 principles of the Agile Manifesto and Agile Leadership Skills

This document is an excerpt from the book:
“Agile Project Management for Government “
Authored by Brian Werham
Published by Maitland & Strong
Reproduced with permission under license

PMCAMPUS.com / Mokanova Inc
.

The 9 Agile Leadership Behaviors

On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, seventeen people met to talk, ski, relax, and try to find common ground and, of course, to eat. What emerged was the Agile Software Development Manifesto.

Jim Highsmith, Agile Alliance

This part of the book takes a detailed look at the 2001 Agile Manifesto and its supporting principles. The Agile Manifesto is presented in Table 1 overleaf.

In the weeks after the Manifesto was signed, 12 related Agile Manifesto Principles were developed, focused on the essential needs of agile teams to run themselves effectively from the process perspective of the development team members.

What I propose in this part of the book are 9 Agile Leadership Behaviors which support and enable these principles. These behaviors focus on the essential needs of the project to be run from a leadership perspective (see Figure 1).

These leadership behaviors are discussed one by one from Chapter 7 onwards. They are presented in Table 2 alongside the related 12 Agile Manifesto Principles that they enable.

Table 1: The Agile Manifesto (reproduced from agilemanifesto.org)

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

That is, while there is value in the items on the right, we value the items on the left more.

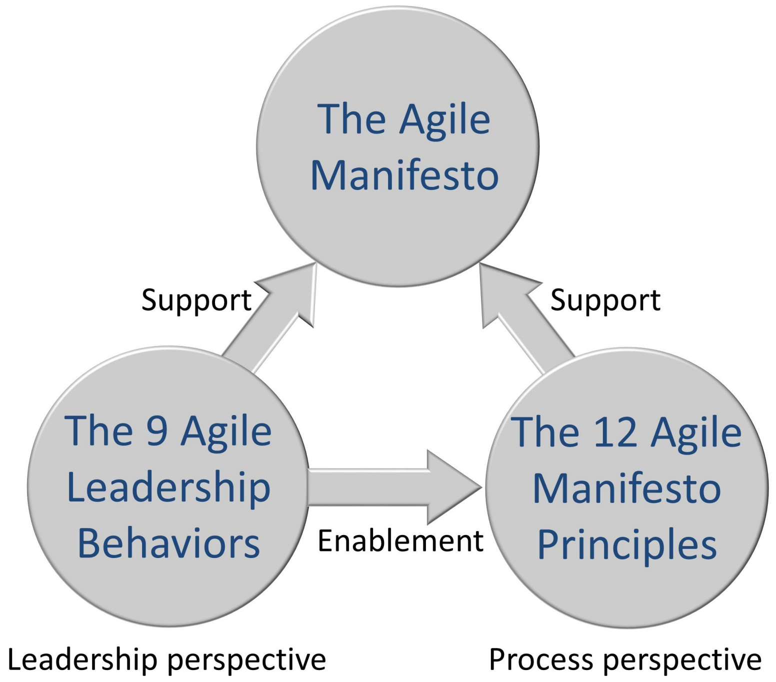

Figure 1: The Proposed 9 Agile Leadership Behaviors Enable and Support Agile Success

Table 2: How the 9 Agile Leadership Behaviors are related to the 12 Principles of the Agile Manifesto

9 Agile Leadership Behaviors	**The 12 Agile Manifesto Principles they enable**
1. Satisfy the customer	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software – Principle One.
2. Harness change	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage – Principle Two.
3. Encourage incremental implementation	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale – Principle Three.
4. Get the business and technical people together	Business people and developers must work together daily throughout the project – Principle Four
5. Create trust through leadership and process	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done – Principle Five.
6. Encourage face-to-face conversations	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation – Principle Six.
7. Set targets and reward real progress towards a working solution	Working software is the primary measure of progress – Principle Seven.
8. Give your teams the space they need to excel	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely – Principle Eight. Continuous attention to technical excellence and good design enhances agility – Principle Nine. The best architectures, requirements, and designs emerge from self-organizing teams – Principle Eleven At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly – Principle Twelve.
9. Pursue simplicity, not complexity	Simplicity, the art of maximizing the amount of work not done, is essential – Principle Ten.

The Agile Manifesto and its Principles

This chapter explores the Agile Manifesto and its Principles and then outlines the need for people at all levels to adopt the 9 Agile Leadership Principles.

Background to the Signing of the Agile Manifesto

Aspects of agile development have been valued by many in the development community for a long time. For example, taking an iterative approach has been recommended in much of the DOD guidance. But the advantages of incremental development were downplayed by the ideology of 'one size fits all' methods in the 1980s and early 1990s (discussed in more detail in Part III).

In 2001 a meeting took place at which a manifesto was signed which reinvigorated those who were worried about the dangers of waterfall approaches. The signatories of the Agile Manifesto placed themselves against the extreme adoption of BDUF, waterfall, and top-down theoretical designs. They knew that processes, tools, documentation, contracts, and documented plans had their place in most forms of organized work at any scale, but they wanted to emphasize the primacy of the collaborations of individuals and the output of working solutions rather than documentation. The Agile Manifesto was very brief: it was only a few sentences long, and more of a statement of intent than workable guidance.

After the meeting, a set of 12 principles were developed to give a more detailed definition and support the philosophy of the Agile Manifesto. The principles have strong support from agilists, but are not well-known outside of those interested in agile as a technique, rather than as a business tool. In researching this book, I found that project failure or success generally depends on top management behaviors combined with technical ability. The Agile Manifesto Principles target the team and their technical environment. They do not explain the role of leadership in encouraging and facilitating agile success. Therefore I have analyzed the agile case studies from the perspective of leadership behaviors rather than the perspective of management processes. The 9 Agile Leadership Behaviors that I have identified and their relationship to the 12 Agile

Manifesto Principles are shown in Table 2.

You may ask why there are three fewer Agile Leadership Behaviors than there are Agile Principles. There are good reasons why the relationship is not exactly one to one. Research by Laurie Williams has identified various small anomalies, duplications and cross-overs between the principles. She undertook a major survey to see how well it and the supporting principles had dated. Of the 326 experts surveyed, there was considerable support for the continuing robustness of the principles. Of the respondents, 90.2% considered the principles as more important than any specific agile practice. 100% of respondents found the principles valuable in themselves, above and beyond the overarching manifesto statement.[i] I agree with her conclusion that although the ten-year-old principles could be polished, they are robust enough for their purpose and well-known and accepted by Agilists. However, since I am now proposing a set of Agile Leadership Behaviors from a management perspective, I have taken the opportunity to incorporate some of her recommendations on agile perspectives in this book.

The Agile Leadership Behaviors I am proposing are typically terser, less repetitive, easier to remember (being only 9 rather than 12) and less internally focused. In other words, I propose them for the purpose of changing the behaviors of politicians, managers, business people, auditors, and procurement staff as well as technical people.

Some have argued that in the new millennium we have entered a *post-method era*. Web development, outsourcing, use of readymade solutions, and cloud-computing undercut the assumption that large complicated methods and procedures are needed to structure the team activities of the developers of technology solutions.[ii] Since the 1990s, interest in highly structured complex and prescriptive methods of development has receded. One study in 1998 found only 6% of private organizations following a method rigorously, with 79% of those not using a method having no intention to adopt one.[iii]

So why the resurgence of interest in methods of software development? Especially agile methods?

The trigger that set the agile revolution in motion was the cancellation of the Chrysler C3 project in February 2000 soon after the ill-fated merger of Daimler-Benz. The techniques that now make up eXtreme Programming (XP) were developed by Kent Beck, who worked on the C3 project as a response to the trend towards larger and larger design-led software development projects. Even though use of complex methods had become unfashionable, projects still depended on the concept of BDUF and waterfall life cycles.

Beck called a meeting of XP practitioners in Omaha to discuss the future of the method. Those at the meeting agreed that there was a lot of common ground between the participants, but no clear solution emerged as to how to make the world aware of these new techniques. In February 2001, Robert C. Martin called a meeting at the Lodge at Snowbird ski resort in the Wasatch Mountains of Utah. Participants included proponents of a diverse number of methods, including Extreme Programming, Scrum, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, and Pragmatic Programming.[iv]

The discussions focused on their distaste for BDUF and waterfall approaches. In the end they found a way of expressing their philosophy, even though when it came to detail they had different methods. They drafted a document they called the Agile Manifesto. It consisted of four short statements in the form "we prefer X over Y".[v] Martin Fowler describes that meeting's objective as the creation of "a rallying cry to the software industry ... it says what we stand for and also what we are opposed to ... a call to arms". [vi]

The manifesto is written in the style of polar opposites – what Beath and Orlikowski identify as dichotomies, each comprising two elements. A *privileged* element is presented in preference to and in opposition to a *deferred* element. They point out that when used to describe development methods, these opposites often prove to be inseparable and dependent on each other. The Agile Manifesto explicitly recognizes this feature by stating "while there is value in the items on the right, we value the items on the left more". If this last sentence in the Agile Manifesto is overlooked, then the statements can be mistakenly interpreted as exclusive dichotomies. They are not. For example, although Agile methods focus on working software rather than detailed documentation of designs, there is still a need for some documents – especially when tracking progress and planning work (see Part II).

The term *lightweight* had been used up to that point as a general description of the different methods that the participants were proposing. That term was discarded in favor of the term *agile*. This was thought to capture the common aspects of the adaptive nature of the related methods represented by the participants, without implying that the proposed approach was flimsy or facile.

So let's examine the four manifesto statements and their implications for large organizations.

Agile Manifesto Statement One: Valuing Individuals and Interactions over Processes and Tools

This statement orientates agile away from the prescriptive, process-oriented structured methods and away from the complex tools needed to follow such methods.

First, consider the preconception of many people that waterfall approaches are inherently superior, if only they are followed strictly enough. At the height of their popularity, these prescriptive, highly structured BDUF methods became widely adopted in governments around the world. The "Structured Systems Analysis and Design

Method" (SSADM) was in use from London to Melbourne, from Ottawa to Hong Kong.

The first large-scale casualty of SSADM was the London Stock Exchange's Taurus project. Its objective was to radically restructure securities trading in London and introduce fully automated cash settlement. But, after six years and nearly £500m expenditure, the project was canceled.[vii] The requirements had been documented using SSADM at a detailed level, but there still was no clear understanding among project staff regarding the interaction of technical, business and institutional requirements. The project was canceled before a single module was implemented.[viii]

Second, consider the fact that the Agile Manifesto distances itself from dependency on complex tools. BDUF approaches may employ expensive and difficult to use Computer Aided Software Engineering (CASE) system design tools. The aim of these is to capture a very detailed BDUF analysis and then automatically generate programs. As teams grapple with the inflexibility of updating the detailed designs before programs are generated, they start to bypass the design activities and simply use the CASE tools as glorified programming tools.

Agile Manifesto Statement Two: Valuing Working Software over Comprehensive Documentation

Agile methods are based on the assertion that decisions must be based on evidence that comes from experience. Documents, however carefully compiled and checked, cannot provide evidence of worth. Agile therefore depends on "transparency, inspection, and adaptation".[ix] Transparency comes from regular demonstrations of the emerging solution and its proof in early live use. Quality assurance by independent teams is enabled by rigorous detailed testing conducted by developers and user checks before a solution goes into live use. There is nothing un-agile about letting an independent team check that mission-critical systems have been tested and are free of fraudulent code. Adaptation is possible because development occurs in short cycles which allow developers to change the design and users to change their minds without either loss of face, or large amounts of rework.

As toddlers, we learn to walk, getting regular feedback on the evenness of the ground under our feet. As children we learn how to cross a road safely, using our eyes and ears to check for approaching cars. As adults, when we are rushing to work, we may consider stopping to buy a coffee when we smell the aroma coming from a café. It is the natural way that we run our lives, and is also the optimum way to run a project: in short iterations that give feedback.

A good business change project is one that not only balances economies of scale against risks of big-bang, but also recognizes the need for feedback from real-life implementation to drive changes to targets. Such thinking is often termed *empirical process control*. Its application to complex business changes came out of Shewing's work on continual improvement of quality in manufacturing processes. He argued that more use should be made of data about the products to adapt and improve processes. This idea was adopted by the Japanese in the search for improvements to their recovering industries after the Second World War, and it was later popularized by Deming as a four-step PDSA cycle (see Figure 2).[x]

The waterfall life cycle neglects the importance of feedback and replanning. It assumes that if enough planning is done upfront, then it will never be necessary to deviate from that perfect plan. This is the *defined process control model*. The Deming PDSA model is an *empirical process control model* – the model emphasizes the need to change the plans regularly using an evidence-based approach. To take full advantage of this theory, we must recognize that:

- ♦ Only an immediate project plan is required in detail – just enough to allow work to proceed to a point where evidence can be gathered on how effective progress is in real-life
- ♦ Evidence must be collected while carrying out tasks – on the effort consumed, the qualities of the outputs, and also on the benefits that the technical solution brings
- ♦ Effort needs to be put into studying lessons learned – could the work have been carried out more efficiently? Were any recurring problems found during testing? Did the resultant business change produce the intended benefits? Were there any disbenefits?
- ♦ Decision-makers spend time considering the evidence.

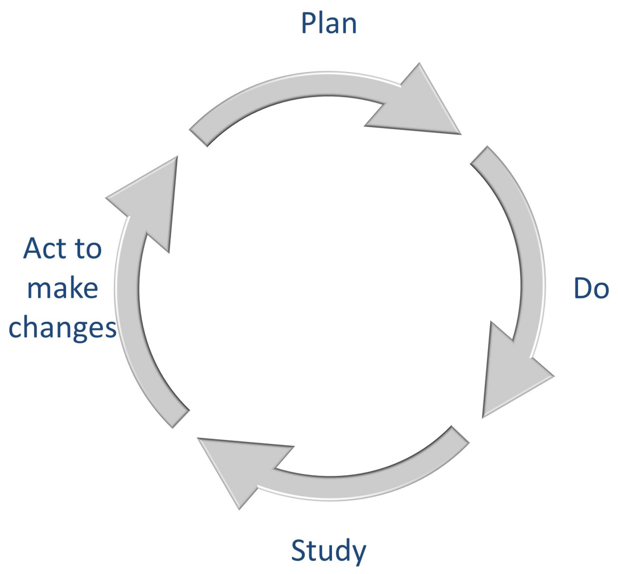

Figure 2: The Plan, Do, Study, Act (PDSA) Model

Most importantly, good leaders accept the inevitability that initial plans will always need to change. Early feedback is needed to incrementally improve the initial overview plans. Techniques such as prototyping, piloting of the solution, parallel running alongside any existing processes, and phased implementation all should be used to provide feedback on the concepts that underlie a project.

Great leaders plan for data to be collected, make enough time to analyze it, and ensure a blame-free culture that allows for easy adoption of changes to plans. At the heart of agile is this concept.

Ken Schwaber makes the difference clear: careful thought is required before choosing to run a waterfall project based on the defined process control model:

> "We use defined processes (in everyday situations where) we can crank up unattended production to such a quantity that the output can be priced as a commodity. However, if the commodity is of such unacceptable quality as to be unusable, the rework is too great to make the price acceptable … we have to turn to, and accept the higher costs of empirical process control. In the long run, making successful products … using empirical process control turns out to be much cheaper than reworking unsuccessful products using defined process control." [xi]

The defined process control model expects a specific output to be produced from a pre-determined, exact process laid down at the beginning of the work. Little or no feedback is built in to the process. With empirical process control, on the other hand, regular feedback ensures that the project keeps on track.

In short, we should use feedback and empirical experience to adapt to changing circumstances. We minimize the cost of failure with an iterative approach: the cost of failure is limited to merely the last iteration of work. And the smaller the iterations are, the smaller the cost of potential failure becomes. If the project is high risk, then this theory of empirical process control suggests an iteration length as short as possible. This model shows how mega-projects lie at the opposite end of the pre-planning spectrum from agile projects.

The main determining factor for optimum iteration length is how much you can afford to write off in case the output is unusable. If regular implementation of the output is not onerous, then very short iterations are preferable. If implementation and actual usage cannot be achieved at each iteration, then a proxy measure of success can be used, such as user acceptance testing.

Agile Manifesto Statement Three: Valuing Customer Collaboration over Contract Negotiation

Mary and Tom Poppendieck have identified several traditional engineering contract negotiation approaches as *anti-patterns* for software project management – in other words techniques that are not just a waste of time, but that actually engender harmful behaviors. They criticize detailed work breakdown planning and bureaucratic scope control. They also criticize the use of *earned value analysis* (EVA). This is where the value of an activity is calculated as being equal to the money spent on it (see the discussion about EVA in Part III). These processes often end up being used by the government and the supplier as a means of gaining an advantage in negotiations, rather than working to find common ground.[xii]

These specific anti-patterns were evident in the problems encountered in the FBI Sentinel. A great deal of emphasis had been placed on technical project acquisition skills and negotiation. But there was not enough collaboration and discussion between the project managers and the technical experts who could see plainly the technical difficulties of the technology that was being proposed. Project managers had been hurriedly trained, certificated, and then placed into top management positions. The technology experts who understood much better the difficulty of the tasks in hand had to take direction from inexperienced project managers. Their training had consisted of either a nine-day boot camp, or, for the more senior, an eight-week course followed by the PMI Program Management Professional (PMP) exam. The key measure of success was the number of new PMPs, not their ability to lead technical teams to success. Experienced engineers with top flight engineering degrees were placed under their command. This had two catastrophic consequences. First, there was no effective collaboration with the sub-contractor from a technical viewpoint, and poor quality outputs were accepted without any root cause analysis or pressure being brought to bear to improve performance. And second, the planning of work veered towards the easier elements, always pushing back the more challenging aspects for later.[xiii]

Agile Manifesto Statement Four: Valuing Responding to Change over Following a Plan

A major side effect of pursuing BDUF approaches on large, public sector projects is that of overplanning.

A false security often comes about when the design of every aspect of a solution is attempted before any part of the solution has been built and trialed. This confidence in the design is usually reflected in overcomplicated plans that provide a detailed narrative for the whole of the project, and in a lack of attention to risks and contingency planning just in case things don't turn out as expected. In the same way that CASE tools were built to support ever more complicated and complete modeling techniques in the 1980s, complicated planning tools were developed that supported the perceived need for detailed upfront planning of a whole project. Barry Boehm has described these *inchpebbles* – making the point that they do not provide a helpful way of knowing whether you are on the right track, as good project *milestones* should. These inchpebbles merely lock all parties into an "ironbound contract" with no flexibility in direction:

> "Excessive, prespecified plans overconstrain the development team even at minor levels of change in personnel, technology, or commercial off-the-shelf upgrades. Such plans also provide a source of major contention, rework, and delay at high-change levels." [xiv]

One study of four large failed software implementations whose costs at project cancellation were between $9m and $112m found long, detailed complicated plans for monolithic big-bang implementation. The contracts based on these plans did not protect the client companies from the risks of failure, but actually helped large consultancy companies extract large fees for longer.[xv]

When a great deal of time and effort has gone into detailed long range planning, this can build a great deal of inertia into a project. Any required change has a great impact on the management products and contracts that have been agreed up-front. The effort required to introduce even the smallest change is enormous, and is resisted.

Of importance are the changes that occur in the target environment. In Part I we saw that initial attempts by the FBI to modernize their case management system failed to take into account the BPR exercise that had been underway for some time.

Civil service initiatives such as the UK Technology in Business Fast Stream for young civil servants and the new Major Projects Leadership Academy for experienced senior civil servants may be able to start to modernize the approach to project management.[xvi] The White House Office of Management and Budget (OMB) has announced a specialized career path for IT program managers, but the job specification only explicitly recognizes a waterfall life cycle. Even when it is taught, there is always the danger that the agile approach will be segregated as a special technique rather than being the default basis for most technology projects.[xvii]

An example of the importance of training in collaborative project management skills is the US DOD. There are over 126,000 military and civilian procurement specialists in the DOD, working in more than a dozen different services and agencies. With good education levels and low turnover, the skills base should be very effective at planning and running acquisitions. About 40% of these staff members work within either program management or systems planning activities.[xviii]

Since its inception in the 1990s, as a result of the Packard Commission review of the management of the DOD, the Defense Acquisition Workforce Improvement Act (DAWIA) has gone through several revisions. The latest revision in 2006 attempted to increase the quality of contract management by requiring the DOD to set up the Defense Acquisition University (DAU) and to establish education and training standards and career paths. DOD-5000 established formal definitions of competencies and career paths for program managers, computer systems developers and auditors, among many other categories.[xix]

Good management practices can enable government departments to manage their own collaborative technology projects. An example of how the often inflexible contract negotiation that occurs with prime contractors can be eliminated is the US Social Security Administration (SSA), which now performs nearly all IT program work using its own people, infrastructure and systems.

The SSA enhanced and focused its internal project management capability in the wake of a memorandum issued in April 2007 by the Office of Federal Procurement Policy (OFPP). The memo required each Chief Acquisition Officer (CAO) to develop a workforce policy to ensure agency project managers had essential program and project management competencies. These competencies laid the foundation for the SSA to take control of their own projects.[xx] Furthermore, the education program now includes specific support for changes introduced under the OMB's 25 Point Implementation Plan reforms to IT in the UK government. By 2011 the SSA had 85 certificated project managers. The SSA is now proud of its project management capability:

> "Although an objective measure of the PM program is difficult, SSA has had no program failures (and few difficulties) since adopting the program. SSA has also been recognized for superior PM competency during TechStats and other reviews/audits and has been sought out by Federal Acquisition Institute (FAI) and other PM leadership groups for input and participation. Of SSA's 17 major investments on the Federal IT Dashboard, fewer than 20% are *yellow*, compared to 37% of major investments government-wide. SSA has no major investments with an overall score of *red* on the Federal IT Dashboard." [xxi]

Conclusions

The 9 Agile Leadership Behaviors introduced in Table 2 are an adjunct to and a reflection of the Agile Manifesto Principles. They come from a business perspective that non-technical people will find easy to grasp and implement. Everybody will find these leadership behaviors useful in discussing and explaining the advantages of agile to others, and influencing them to support agile adoption. People who exhibit these behaviors will enable and facilitate agile success, even if they do not know a great deal about the detail of specific agile methods.

If you follow these 9 Agile Leadership Behaviors, you will support your teams in their adoption of agile methods and sticking to the 12 Agile Manifesto Principles in the running of their projects. It is not enough just to train the technical staff in an agile method, if waterfall leadership behaviors still abound.

---

[i] {Williams 2012 #239}
[ii] {Avison 2003 #103}
[iii] {Fitzgerald 1998 #102}
[iv] {History The Agile Manifesto 2012 #107}. Three other experts also attended who brought different methodological perspectives: Alistair Cockburn, Jim Highsmith, and David Thomas. Cockburn had just written a book on the use of diagrams to describe user requirements, Highsmith a book on collaborative development, and Thomas was working on UML-based notation.
[v] {Thomas #108}
[vi] {Fowler 17/01/2012 #106}
[vii]£80 million spent by the London Stock Exchange and at least an additional £400m million by securities companies {Drummond 1996 #125}.
[viii] {Drummond 1996 #125}
[ix] {Schwaber October 2011 #115: 4}
[x] {Shewhart 1931 #42: Dedication} and {Imai 1991 #41}
[xi] {Schwaber 2004 #44: 3}
[xii] {Mary Poppendieck 08/11/2003 #147: 8}
[xiii] {Israel 2012 #383: 76}
[xiv] {Boehm 2002 #132: 65}
[xv] {Vogt 2002 #134}
[xvi] {NAO 2011 #131}
[xvii] {Berry Dec 2011 #162} and {Kundra 2010 #157: 13}
[xviii] {RAND Corporation 2008 #401}
[xix] {RAND Corporation 2008 #401}. For the purposes of explanation and brevity to the general reader I have used “DOD-5000” as shorthand for the series of “5000”guidance materials. Please see {US DOD #434} for the complete index. Don’t forget to notify me if any links are broken, and I will attempt to send you the correct link within 30 days. eBook owners can turn on automatic errata correction to receive updated versions of the click-through endnotes.
[xx] {OMB 2007 #366: 1} and {CIO Council 2011 #367}
[xxi] {CIO Council 2011 #367: 4}