

# Costs of Compliance: Agile in an Inelastic Organization

John J. Cunningham  
IBM Software Group  
*john.cunningham@us.ibm.com*

## Abstract

Doing agile development in a relatively inelastic environment, where policies and procedures are virtually unchangeable, creates an impedance mismatch between the agile team and its host organization. Our experience on a variety of embedded Java projects has shed some light on the costs of complying (or failing to comply), where we trialed everything from “refusal to comply” to “full compliance”. Regardless of approach, there was always an associated cost, whether in redrafting documents, reducing functionality, spending time in meetings, losing focus on deliverables, or deteriorating morale. In keeping with the Agile philosophy, when our efforts were failing, we refactored our approach to managing the project in an effort to minimize the costs of compliance without adopting more risk to ensure success. In the end we were faced with the questions "How did we fare in the end? Which costs were worth bearing? Was it all worth it?"

## 1 Background

In 1994 when I first joined Object Technology International, I started a small services team with another software engineer and we called ourselves the Application Architecture Group (AAG). Our mission: to provide mentoring and project assistance to partners and customers adopting our Smalltalk software technology and products. We concentrated on project startup and project closure – the two periods in which entropy is highest and people would benefit most from our assistance. Two years later, we reformed ourselves as the Applications, Consulting, Mentoring, and Engineering (ACME) team. Our mission was substantially the same, but we added some custom engineering outsourcing capabilities to our bag of offerings.

After being acquired by IBM a short time thereafter, we began to focus more heavily on the embedded software marketplace, which was beginning to toy with object-oriented software concepts and tools. Although we did a handful of Smalltalk embedded projects, it was clear that the future would lie with Java, so we retooled

and embraced the new language and tools offered by IBM as VisualAge MicroEdition, and later Eclipse.

Within the IBM Software Group, we are known as the Embedded Java Enablement Team (eJET) and have been charged with assisting early adopters of IBM’s embedded Java technologies, very much as we had previously done as AAG and ACME. Although our area of focus has been mainly in the automotive and telematics, we have recently expanded into applications utilizing radio frequency identification (RFID) for supply chain management and remote asset tracking, monitoring, and management.

Regardless of industry, our project techniques have remained consistent over the years. Since the early 1990’s, we have applied a Workshop Driven Development (WDD) approach for all of our engagements and internal product development efforts. Since we began this long before the ‘official’ emergence of an Extreme Programming or Agile Development community, we have *rolled our own* development techniques. However, our practices overlap considerably, since members of our team have much in common with, and have had professional interactions with, many of the leading lights in the Agile community. For instance, we practice pair programming, we do test-driven development, we refactor voraciously, we use timeboxes for planning, we gather use cases to drive our test cases, we produce continuous releases, we discover architectures through implementations, we value people over processes, and working code over documentation.

Our track record is one of repeated success in demanding environments. Passionate about creating working systems, we drive our projects through application of scientific methods of inquiry to our development efforts. We define our problem, posit a solution (hypothesis), attack the points where our hypothesis is most vulnerable (by coding tests for validation/proof or counterproof), assess the tests, alter the hypothesis in response, and then retest until we converge upon a repeatable solution.

## 1.1 Workshop Driven Development

We named our method of software engineering, *Workshop Driven Development*, because our development efforts are punctuated by a series of workshop sessions that help us to achieve common understanding, vision and convergence on project objectives.

We begin with an Opening Workshop when approached with a problem. At this initiation point, we set out to first work with our customers to establish the common problem statement in the form of a collection of use cases. To us, a use case is essentially a high level narrative of WHAT the system must do. We capture a simple process flow, pre-conditions, post-conditions, assumptions, risks/issues, and some notes on important exceptions and dependencies. In the embedded space, we also capture information about attached devices, platform considerations, and critical performance requirements. We also begin the development process during the Opening Workshop, by establishing a common development environment and walking the customer through the necessary bootstrapping to set up the test harnesses. Most critically, we begin engaging the customer in test-driven development of the simplest use case. By the end of this first week, we have a kernel of the system up and running as an example of how to proceed.

Following this first week, we assess the scope and engage in ongoing planning, which includes the layout of a set of Timeboxes. A Timebox, taken from the Dupont methodology [1] of the late 1980's, is a fixed block of time (30-90 days) in which our team undertakes delivery of a set of function (stated collection of use cases). The notion is that resources (i.e., team, time, funding) are fixed, but that the functional content may vary. To make this work, it is critical to satisfy a set of checkpoints prior to starting a Timebox. A checkpoint can be scope validation, background or input document availability, personnel availability, contracts and funding in place, component or hardware availability, technology availability, agreement on priorities, and/or review of prior timebox success. Within a Timebox, we generally schedule one workshop in each 30 day period as Development Workshops, during which the team gathers together to attack the hardest problems we face or to focus on specific skills transfer.

All of this is wrapped up by a Closing Workshop, during which we provide demonstrations of the delivered systems, review our planning accuracy, and obtain customer acceptance and sign off. During this workshop, we normally begin discussions pertaining to future work, either as an extension to the content delivered or related to some new opportunity.

## 1.2 The Inelastic Organization

For the most part, the members of the IBM Embedded Java Enablement Team previously worked in smaller organizations, where processes were generally much lighter weight and organizational structures were much flatter. However, we also have specific experiences either directly in large corporate organizations, or with large corporations as customers. This exposure to a variety of organizations enabled us to recognize that the larger organizations often develop very specific prescriptive processes driven by their perceived necessity to scale and by a belief that firm processes increase predictability and reduce risk.

Our prior experiences in smaller organizations taught us to adapt our processes and methods to change to survive and grow. On the other hand, we also observed that reliance on rigid processes could foster resource interchangeability and stability. Demand for conformance to process standards corresponds often has a correlation with the maturity or scale of the business. Large, mature businesses have established protocols and strict guidelines and demand compliance. Smaller, newer organizations are nimbler and more responsive to change as the feedback from adapting appears more direct. I characterize an organization's ability, or willingness, to change and be flexible as its "elasticity".

A fully inelastic organization will demand compliance with all standard operating procedures, whether defined explicitly or embodied in behavioral norms. Straying from that path is viewed as organizational deviance or insubordination, and has consequences. The notion of organizational inelasticity is not necessarily a trait woven into the whole of an organization; it may only apply to a department, a division, or a geographic area. Our recent experience with an inelastic organization demanded that we follow a waterfall development plan, participate in daily meetings and conference calls to discuss status, to produce standard sets of documentation (even at the expense of completing working code). For a team accustomed to working in smaller, nimbler organizations, this was viewed as overhead activity that did not appreciably contribute to delivering working systems. The situation embodied the classic clash of emphasis captured in the [Agile Manifesto](#).

Senior management at IBM has recognized that the nature of the business venture often dictates a need for a difference in approach. Bruce Harreld, IBM Senior Vice President of Strategy, was quoted in an interview on Emerging Business Opportunities:

*"There's no question, it takes the right kind of people, not just for management but for the whole team. It's not for everybody. It's exhausting, there's a*

*lot of uncertainty, there are failures. People who are confident in their abilities, and interested in applying what they've learned to a set of interesting, fun opportunities and in growing personally from it - they are going to thrive. Other people work better within more structure and predictability. It reminds me of my kids: two of them are very structured, and comfortable in a highly controlled environment. My other two are more aggressive and experimental.”*

Unfortunately, business opportunities often arise with an urgency which prevents the organization from profiling team members ahead of time to ensure a proper match. Customer satisfaction and revenue generation take priority, and rightly so. As a result, in the early stages of pursuing a new business opportunity, you may have a mixture of people from both categories: structured and experimental. This is precisely the situation we found ourselves in last summer.

### 1.3 What We Wanted to Do Differently

*“No crime is so great as daring to excel.” – Winston Churchill*

Given the challenge of customer satisfaction in all contexts, our team wanted to apply the best practices that we accrued through successful prior experiences. As a cohesive team with cross-disciplinary capabilities, we were confident that given the requisite authority and responsibility, we could manage another project save. There were a number of things that we insisted were important for us to do, but that were practically at odds with what our more structured host organization demanded of us on a daily basis.

**1.3.1 Focus and Clarity.** We desired to focus exclusively on the application of our people, skills, interactions and knowledge to defined use cases. Anything that did not further our accomplishment of satisfying a demonstration of the requirements or removing a roadblock to doing so was deemed a distraction. Distractions were to be ignored and avoided, since we had clarity of purpose and a means of assessing whether our activities were productive – did they improve our ability to run our test cases and satisfy the use cases.

**1.3.2 Meeting Avoidance.** One aspect of our obsession with focus was to avoid meetings. The only meetings that made sense to us were the planned and scheduled workshops, brief event driven meetings to resolve requirements questions or third-party dependencies, or impromptu quick calls to solicit feedback on technical ideas to solve problems. We avoided regularly scheduled blocks of time for status review, since it was well known that the status review always expanded to fill the time allotted. Besides, we were all well aware of the status at any given time, because we were engaged in continuous

integration. As the project manager, I was responsible for summarizing status on a daily and weekly basis for external consumption. We did have a somewhat arrogant and circular kind of reasoning regarding meetings: if you needed a meeting to find out what was going on, you must not be engaged in productive work – and if you were not engaged in productive work, no one who was would have time to spend explaining it to you!

**1.3.3 Measurement by Working Tests.** Our measurement system was a simple one: how much of our set of test driven scenarios was complete. Rather than break the work for a given scenario down into a set of activities and tasks to be completed, we had shared responsibility to complete the scenario as a team. This coarse grained measurement system enabled us to focus on building our working code while reducing the time and complexity of tracking individual activities. With eleven essential functional use cases, we could say with certainty when we were functionally complete on any given one. If not completed, we knew what was missing, because that was the cause of our test cases failing. In the end, that’s all that really mattered to the customer, so our progress measurement system mapped to our customer needs.

**1.3.4 Lightweight Planning.** We planned only to the level of detail that was required to start solving our problems. We would have impromptu meetings for 10-15 minutes whenever additional detail was required to coordinate between team members. The project manager was continuously engaged with and communicating with team members to elicit from them what resources were required or roadblocks were blocking them from moving ahead as fast as possible. We only looked far enough ahead to the completion of the current timebox, as a team. However, as the timebox stabilized and approached closure, the project manager could begin outlining the planning for the subsequent timebox and validating the checkpoints required to begin it.

**1.3.5 Multidisciplinary Team Roles.** While individual team members were focused on serving in a primary capacity (architect, developer, tester, project manager, or analyst), it was common practice for each of us to pick up the role of someone else when necessary. The team was ego-less and non-hierarchical, driven by a group desire to deliver the scenarios. Everyone wrote code. Everyone wrote test cases. Everyone edited or updated documents. Everyone engaged in architecture and design discussions. It was a “one for all and all for one” atmosphere. The project manager did not direct the team in accordance with a plan; so much as facilitate the team’s accomplishments in accordance with the goals.

## 2 Refusing to Comply: The Take No Prisoners Approach

*“Courage is rightly esteemed the first of human qualities...because it is the quality which guarantees all others.” – Winston Churchill*

In the summer of 2004, we faced a situation in which our team was aggressive and experimental, but our host organization emphasized structure and control. There was an impedance mismatch between the two that needed to be resolved, one way or another.

### 2.1 Into the Burning Building

Our team’s specialty has developed over time as the team that people come to when a project is in trouble. June 2004 was one such instance. An RFID customer project scheduled to go live in November was needed help. Aware of our reputation, an executive visited us and offered us a challenge: get this project shipped and great rewards would be ours. The next week, three of us were on a plane to Europe to see just what we had been volunteered to do. Upon arrival, we met the local project team and began to take the measure of the state of the project – what assets were in place, what code had been completed, what scope had been committed, and what acceptance criteria had been agreed upon.

After three days of intense conversations, the risks seemed insurmountable. As currently understood, the project could not succeed – the observed risks were too great and we had no leverage to control them. The core problem was that the project did not appear to be driven by a stated business objective, but rather by a common fascination with a new technology. Our experience is that this is a common reason for projects to arrive at the state where they seek our assistance. There was no clear answer to the question, *“What business problem are we trying to solve?”* In order to make this project successful, we needed to radically redefine the project scope and introduce extreme efforts – particularly with respect to resetting customer expectations.

The customer had not aligned his internal constituencies, so we were trying to satisfy competing agendas within the context of a single project. Additionally, in our brief discussions with the customer, he could not clearly and concisely define his own business processes – they mingled what existed with what they wanted, causing confusion. With regard to the new technology introduction, the underlying RFID technology (independent of software) exhibited brittleness in radio physics (i.e., sensitivity, detection, interference, etc.) and the existing demonstrations were unconvincing. Furthermore, the selected timeline seemed inconsistent with our capability for delivery of numerous components

– vaguely stated goals are unachievable.

Too many of what normally would be the ‘simple’ knowns this close to delivery remained unknown: hardware selection, non-functional requirements, functional requirements, deployment strategy and precise scale of rollout. No acceptance criteria had been defined for November, therefore there was no basis for measuring remaining work, let alone for assessing or declaring success – a very bad position to be in. Too many control points were beyond our control: hardware availability, platform integration and contracted support. We had hard, critical dependencies on third parties and they were feeling the same constraints.

Our interactions with the project team were constructive and welcomed. Some of our observations and advice reinforced their own private assessments, so there was significant agreement. Admittedly, our American contingent may not have been as *nuanced* in expressing some thoughts – but our candor and willingness to assist was appreciated. We established open, honest lines of communication and focused together on solving the remaining problems, choosing not to concern ourselves with how things had gotten to this point. At a minimum, we provided a sanity check for the project team and together established a common set of objectives.

### 2.2 Gaining Control

Our first order of business was to make a run at gaining control of the project. The last thing I wanted to do was engage the team in a death march to November. The unspoken assumption was that this is precisely what was in store, and it hung heavily on the existing team’s shoulders. Therefore, we defined a small scope for delivery in July to test our ability to control the content and satisfy the customer. In order to meet this test, we scoped ourselves to just four use cases, two functional and two non-functional:

M101 – Optimistic Pallet Tag Reading

M102 – Ignore Case Tag Reads

M201 – Run Automated Functional Regression Tests

M301 – Transaction Time Trials

Our thinking was simply that if we could get the most optimistic scenario implemented end-to-end and then complete a variation on it that filtered out case tags, we were on the right path. Furthermore, we needed to construct the automated test harnesses necessary for our test-driven development models to ensure quality and repeatability. Additionally, our customer demanded to see that the total transaction time from initial tag read to visual cue, would be under the two second threshold.

The first step was to obtain clear use cases for each of the four scenarios for this initial Timebox. We took a first pass at writing these up ourselves, and then handed them over to the local project team to validate and complete with the customer. We then encouraged the local project team to get the acceptance criteria resolved with the customer, in writing. It was imperative that we cleared the first hurdles of knowing what was to be delivered and under what terms it would be accepted. Our partnership with the local team was strengthened as we worked through these issues. We rapidly developed a shared objective and understanding of how success and failure would be defined. Up to this point, everything was going very smoothly. The teams had embraced one another's views, established a common set of objectives and were actually excited about the challenge.

### 2.3 Getting Started

The next step was to execute and deliver. We applied our proven *Workshop Driven Development* approach. We gathered three key members of the local team in our lab in North Carolina to begin the development effort with an Opening Workshop. We had the clearly stated use cases, people with domain expertise who had worked with the customer and our team who were skilled in the embedded Java technology. We needed to spend a week together getting as far as we could, using simplifying assumptions, on each of the four July use cases. Our plan was to spend the week building some working models and then reflect on them to answer the following questions:

- Can we write our top level test cases from the use cases?
- Can we make something that works – satisfies the test cases?
- Does this working model point to a specific architecture or design?
- What have we learned about the level of effort required to add new function?
- Can we now put together some estimates and plans for completion?

Although we had a proven track record of success with this approach since 1994, not everyone was familiar with us. On the second day of the first workshop, we began to get the first requests for “The Plan”. I responded by candidly confessing that there was still no plan since we were in the Opening Workshop and needed to complete our week's work first. We intended to have a draft plan for the Timebox the week following this workshop. Apparently, that was the wrong answer, because the demands for a plan became louder and more frequent. My management was incredulous that I would

embark on the project without a plan. I was asked: “How will you know when you are done?”, “How can you determine how much work remains?”, and “How do you know who is working on what task?” My answer was that we still didn't really understand enough about the problem we were solving and needed the week's workshop to make that assessment. I explained that any plan created in this knowledge vacuum was worthless and would be a throw-away, at best, and a source of frustration for managing change and taking our focus off the objectives, at worst.

On the third day of the workshop, we had the optimistic case running in a full test harness with simulations for all of our hardware peripherals. We had an approach in mind for how to handle the case tag filtering and had one member working alone on the side to build a monitoring tool to measure transaction times for an instrumented version of the code. For those of us who had been through these workshops, we were relieved and happy that things were going so well. We had refactored three times already, twice on Day 1 and once on Day 2. We settled on an extensible programming model that was clear, that everyone understood, and we all had faith in.

As the project manager, I must be capable of understanding the software that my team is creating – willing to open up a browser and read through the code to grasp the function and organization. It is quite normal for me to open a browser in a closing workshop to direct a code walkthrough in order to demonstrate to a customer that our deliverables actually meet the requirements in detail. Therefore, I intended to spend all of my time that first week participating in the technical workshop as a party to the test, build, design, and architecture discussions and decisions. After all, with that depth of understanding, I would then be in a much better position to leading the planning activities.

However, we still didn't have “The Plan.” Our willingness to throw away a half day's work and do wholesale refactoring so often was viewed as evidence of our “lack of direction”. I was told in no uncertain terms that a plan must be completed immediately and that the project would not be done this way. I simply refused to comply. I argued that we were selected to undertake this project specifically because of our skills, approach, and track record. I pointed to our progress as evidence that we were on the right path:

- We had working code in just three days!
- We had unified objectives and a challenging goal
- Our team organized in a results-oriented manner
- We established a collaborative climate

- We had competent team members from different groups
- We had agreed on standards of excellence
- We had principled technical and project leadership

In fact, the only thing we lacked in the eyes of Larson and Lafasto [2] to complete a high performance team was external support and recognition.

## 2.4 Fallout

Our department was accustomed to having plans, or checklists, of activities and their status. My refusal to comply with that convention tagged me as insubordinate. The frustrating aspect of this was that I had previously been assured that we would be granted support and allowed to employ our Workshop Driven Development approach. I repeatedly insisted that those were the terms under which we agreed to take on the risky project, and that I would not back down. Not only was the principle involved, but I also knew that it would have serious detrimental effects on the balance we had achieved in forming the team. The situation went from bad to worse. Even before we had completed our first workshop, another project manager was shadowing me and trying to build a “standard project plan”. Even though I had assured them that I would have a plan completed early the following week, it was not enough – I was rogue, operating outside normal controls.

In order for the shadow manager to build the plan, she needed to interview team members and find out what tasks needed to be done and how long they would take and who needed to do them. They were being pulled out of the workshop for these discussions, thereby slowing our progress. My authority as project manager was being undercut by these activities, so I confronted the other manager and asked what was going on. I was told point blank that since I refused to provide the required Plan, someone else was going to do it. I appealed to the project executive to formally restore my authority. He agreed to intervene and I was granted until the following week to complete planning, which was then done.

Throughout this early phase, it became clear to me that our priorities were shifting. The plan was becoming the deliverable of primary importance, rather than our working code. Again, the other project manager was dispatched to gather specific planning information and again began to take time away from the developers who I had tasked to deliver code. Instead, they were spending their time trying to explain what they were doing and how it all related to the final deliverable. Now, because I had participated in the workshop at a technical level, I understood the system in a way that enabled me to plan and document with minimal interference with the

developers. My desire to maintain the integrity of our development methods and my refusal to comply with the documentation and planning demands of the organization were now negatively affecting everyone. While I thought that I was protecting the team from waterfall processes by pushing back, the organization was simply doing an end run around me and asserting its dominance. I asked the team to direct all status and planning questions or requests to me so that they could remain on task. At this stage, I was beginning to think about ways to ‘refactor’ our team’s behavior in order to address the changes around us.

We fended off the distractions mainly by ignoring the requests for information to complete the organization’s form of “The Plan”, instead first focusing on delivering the four use cases and supporting architecture and design presentations for the customer’s acceptance. We succeeded with our July customer deliverables and the customer now had confidence in the November ship date. However, even this major victory did not relieve the pressure of completing “The Plan”. In our practice, we make a distinction between “The Plan” as a motive force, and “Planning” as an evolutionary activity in response to increased knowledge, experience, and change. Normally, we plan and re-plan continuously. We do not create a “Plan” and treat it as some Immutable Law to be obeyed under penalty. So, shortly thereafter, I began planning our next set of Timeboxes with more detail, spelling out the work to be completed through the November delivery. I included the information pertaining to scoping, checkpoints, resource requirements, milestones, and deliverables. However, it was not delivered in a familiar format, and was therefore deemed unacceptable.

Even though we were delivering, we were not conforming to the organizational standards. We desperately attempted to make our case that what we were doing was effective and would continue to work. After all, in just five weeks we had eliminated the major risks of failure, earned goodwill from our customer, and established a firm architectural foundation on which to complete the project. Again, I was informed that the project would not be managed using Timeboxes and a workshop driven approach. I challenged my management at this point to either allow me to continue managing the project this way or to find someone else who would do it their way. By the end of August, I had been officially replaced by a traditional project manager.

## 3 Keeping Up Appearances

*“A fanatic is one who can't change his mind and won't change the subject.” – Winston Churchill*

While in the short term, poking the organization in the eye with a sharp stick by refusing to comply enabled us to

meet our July objectives and keep the project alive, it had serious consequences. Team leadership was beheaded (figuratively), and a rigorous process for tracking and documenting status was instituted that burdened the developers. We were on track for a classic death march. Processes and tools took precedence over people and interactions. Comprehensive documentation became more important than working software. The new project manager began focusing on following “The Plan”, rather than engaging in planning. Confidence began to erode and the sense of mission began to slip away.

### 3.1 The “Stuffer”

I had been “officially” replaced as the project manager, but I continued to support my team behind the scenes as I was also preparing for the next project. I continued to do so until we had completed implementation of all of our functional use cases, most of our non-functional performance targets, and many of the critical systems management use cases. What remained was essentially tracking of remaining open issues, synchronizing third party suppliers, and bug tracking. A set of status meetings were set in which many people gathered in a conference room, or remotely by phone, to track through the list, line by line, for the most up to date status. The development team was using Bugzilla already to track these issues in an asynchronous fashion. However, they were continually asked to participate in the management meetings to provide the most up to date status possible.

I had continued my dialog with team members over these weeks. Morale was devastatingly low – which was significant for a team with such a long history and high energy level. In our internal discussions, we decided we needed to refactor our behavior and team organization. We needed one person to step up and take on the role of single point of contact for the status and planning meetings. We called this person the ‘Stuffer’ – the person responsible for “*all the stuff that isn’t critical for delivery, but that the organization wants*”. One obvious trap to avoid was having the Stuffer become a proxy for the project manager, and continue imposing demands on the development team. So, the Stuffer had to be knowledgeable enough about the solution to keep abreast of the Bugzilla entries and then engage developers informally during coffee breaks, lunches, and happy hour discussions for clarification. In this way, the Stuffer acted as a proxy for the developers in interfacing with the project managers tracking the lists.

The Stuffer’s activities then consisted of attending the meetings and conference calls, which generally consumed two days per week, and converting information from Bugzilla into standard formats. The compiled

information had to be handled with care. Any time a high risk item was added, it would inevitably spawn a new, specific conference call to address that risk. The unfortunate thing was that these meetings were typically attended by the people nominally responsible for “managing”, none of who had hands-on experience with the technology or issue. This meant that no actual resolution of the issue was possible on the call. The resulting fire drills quickly trained the developers to begin hiding information and from management. They began to do their own distributed project management.

### 3.2 The Project Manager as Stuffer

Having learned some valuable lessons in the RFID project, we embarked on a new telematics project in October 2004. We learned that defiance was counter-productive, but that the role of the Stuffer was useful. Having a project manager and a Stuffer seemed excessive, so we tried to add the Stuffer responsibilities to my project manager basket of obligations. This time, when asked for a project plan at the outset of the project, I offered a standard waterfall-ish plan. Of course, we knew it was wrong, since we hadn’t done our initial workshop and had no real sense of how much effort was going to be required to complete the listed activities. The important thing was that there was no conflict about a plan and we were still empowered to deliver code at weekly intervals. I made an effort to not allow the Stuffer role to consume more than 1 day per week of my time.

By the time we were a couple of weeks into this new project, it was clear that the project planning charade was going to have to be reworked. So, in our weekly status meetings we would first identify the things we had completed and show that we were on track to deliver, but only if we made certain adjustments in the content of our weekly deliverables. I started making changes to the plan, which for the most part were acceptable. After all, it is widely experienced in inelastic organizations that plans break, schedules slip and costs run over. What we were offering was cleverly disguised changes to the plan in order to maintain the schedule. We were keeping to the timebox sensibility, but not calling it that.

Much of the plan alteration was easily attributed to blocking items which prevented us from working on the next planned scenario or deliverable, as outlined in the original plan. As project manager, I could cite the blocking issue as a reason for moving resources onto other activities in a more dynamic, agile manner. The development team was then able to work on scenarios that were unblocked while I went off and worked out the blocking issues with the stakeholders.

This project was a sixteen week effort to build a

Telematics reference implementation for a customer on another continent. Therefore, we could comfortably maintain our workshop punctuated schedule without surprising anyone. In the end, we were able to deliver significantly more functionality than originally called for under the contract in the time and with the budget originally allotted and planning was not contentious.

## 4 Compliance and Its Costs

*“Let our advance worrying become advance thinking and planning”. – Winston Churchill*

In a recent project, much smaller in scope and complexity, I have tried the full compliance approach at the outset as a comparison to our earlier experiences. This current project is more politically sensitive in nature and cuts across the organization. The sense of urgency that existed in other customer projects is absent and the technical challenges are less demanding. Much of that can be attributed to the lack of an external customer to date and to my role primarily as an architect, rather than as a project manager responsible for delivery.

At some point in the coming weeks, some hard decisions will need to be made to bring the project more into focus. In the background, I have been preparing the necessary artifacts to get a small team off to a running start when that time comes. However, at this time sharing that information would be perceived as politically insensitive to others and overstepping my responsibilities. So, we play the waiting game to align with the organizational priorities as they currently stand. Full compliance to date has meant that we have only documents and no test cases or working code. However, everyone seems quite content on the conference calls and in the meetings with our progress.

### 4.1 Our Costs of Compliance

Regardless of the approach taken, compliance placed a burden on our agile development team. While many of these costs began to show themselves at the start of the refusal to comply stage, they became significant when the team leadership changed.

The resulting burden had diminished our resources and taken them off task in the past. Since we used the agile processes and Timebox planning to get end to end scenarios running early, we were more confident early in the project. As attention was redirected toward non-essential deliverables, some corners had to be cut in the product deliverables. The resulting diminished functionality could be attributed to having to feed the process compliance beast, rather than to a failure of skill or understanding on the team's part.

### 4.2 Loss of Focus and Reduced Productivity

The first casualty of compliance in the project was a loss of focus. When the importance of internal documents began to consume a disproportionate amount of the team's time, then the time and effort dedicated to building code diminished. More time was spent in meetings discussing problems than was spent working to solve them. Productivity began to suffer, as the time required to transition for a meeting mindset to immersion in development activities began to increase. Teams don't receive feedback from PowerPoint slides, but they do from JUnit. Loss of that powerful feedback loop hurt.

### 4.3 Team Morale

A form of fatigue began to set in on the project team. Meetings were stressful, because they felt counterproductive. Developers were attending meetings when they felt that their time was better spent fulfilling test cases and integrating new components. Process compliance led to activities which developers felt was either unproductive or counterproductive to the task at hand. Frustration levels increased and more time was devoted to offline “bitch sessions” and trying to formulate ways to avoid being drawn into the process grinder.

The team members were not accustomed to failing to get things done in the time that they had expected, but they were losing time and focus and slipping. At first, work days began to get longer as people tried to compensate for the lost time. People developed short fuses on hard issues and allowed frustrations to boil over. Whereas the office had been an exciting place to be, it was increasingly a frustrating place to be. Work days got shorter, because people didn't want to be there. They would work at home when possible to be isolated from perceived nuisance and conflict. It was also an effective meeting avoidance tactic. What was once a passionate desire to pair program and openly debate design choices has waned in the absence of that close collaboration. Team members more willingly accepted average solutions to problems, because it was easier than stirring up controversy. Isolation and division of labor has since become more common.

### 4.4 Impaired Communication

The most important casualty of enforced compliance on the team was the quality and frequency of communication. It did not take long for people to recognize that communicating anything less than positive news caused increases in meeting length and frequency and imposed additional process burdens. The reaction was to reduce the communication to short statements of accomplishments and to briefly summarize ongoing

activities, and to hide any issues that might cause alarm. As a result, the management team was no longer fully aware of the project risks.

This amounted to a form of institutionalized deception, in a sense. Clearly, open disclosure of status was driving non-productive organizational behaviors. In response, the developers simply altered the information to minimize those behaviors. They are smart people and believed that the true measure of meaning was only whether the project functionality was completed on time. The risk was that if it had failed, management would have been surprised that there were so many open issues “under the radar.” In effect, the developers had taken on the risk management obligations that properly resided with management.

#### **4.5 Professional Well-Being**

Those among the team who pushed back were quickly tagged as insubordinate malcontents unwilling to work within the system. Those who bit their tongues were rewarded with heavier burdens and greater frustration. Eventually, apathy set in and the staff members began openly discussing looking to work on projects in organizations where agile was the rule rather than the exception. Rewards were given to those who conformed to the processes, while those who resisted were reassigned to other departments or told to try harder next year. This did not necessarily align with the value the individual brought to the project. No longer does anyone feel inspired to achieve – but only to survive.

### **5 In Summary**

As much as these experiences frustrated those of us who attempted to apply agile techniques in an inelastic organization, we did accomplish our project objectives nonetheless. That frustration and the consequences earned were the tuition paid for this learning experience. I'd like others to learn these lessons only for the price of reading this paper.

#### **5.1 Quality Results**

The development team bonded tightly with the test team in a way that was previously unknown in the organization. The test team engaged with the development and customer teams very early in the development cycle. Very little distinction was made among the team members concerning such roles – they all considered it an extended team. This dynamic was actually initiated by the same people who insisted on conformity regarding the Plan. The test team was engaged with the continuous builds and never had to deal with the development team throwing code “over the wall”. There was true collaboration, helped considerably by the test-driven approach which guaranteed that the test

team received code of good quality. The test team could then focus on the operational and deployment issues associated with releasing a product, rather than catching the development team's mistakes. In the end, together the team delivered a product that had two orders of magnitude fewer defects than other products in the division. In fact, the handful of defects was well isolated and known to be dependent on a component beyond the project team's direct control. It was an impressive accomplishment and was noted by division executives. Unfortunately, our business unit engaged in reorganization before any of the lessons learned could be institutionalized.

#### **5.2 Non-Compliance is Not an Option**

The bottom line is that in a large organization, whether we like it or not, some degree of compliance will be required. Ideally, you need to find an executive sponsor who will provide the necessary “air cover” that will allow the Agile team to operate with minimal compliance overhead. This is not a right to be demanded by the Agile team, but a privilege to be granted as the reward for proving the capability to deliver. A junior executive may not have sufficient clout to provide the necessary support, so aim high.

#### **5.3 Everything is Refactorable**

Typically, we think of refactoring as the reorganization and refinement of our designs and code. However, the concept also applies to our organizations, our techniques, and our behaviors. When an approach to solving a problem ceases to bring about the desired results, we need to reassess what we are doing and change. This is equally true in how we interact with others in an organization. We cannot be dogmatic about Agility.

#### **5.4 The Role of Project Manager**

The source of conflict in this project stemmed not only from the different development approach used, but also from the perceived proper role of the project manager. In our Workshop Driven Development project model, my duties as project manager begin with the following:

- (1) Set objectives for the team and define clear criteria with the customer for acceptance/success,
- (2) Obtain resources necessary for the team to work optimally,
- (3) Clear obstacles that are preventing the team from working optimally,
- (4) Make decisions on functional triage within timeboxes, and
- (5) Communicate progress and results.

However, the role of the traditional project manager preferred by the organization was to serve as an information aggregator and conduit that creates and directs implementation of The Plan.

## 5.5 Anecdotal Only

It is important to note that the experiences reported in this paper are anecdotal in nature. Our experiences do not reflect the experiences in all of IBM, which is a very large and diverse organization. In fact, we have had experiences in prior years in which IBM supported, enabled, and even embraced our agile approach to development.

Organizations are collections of individuals and as such, often take on the personalities and values of those who direct them. As noted in the earlier quote attributed to Bruce Harreld, there are executives who recognize the value of enabling different approaches in their appropriate contexts. Likewise, when our quality metrics were observed, our immediate executives took note and engaged us in discussions on how to share those lessons and tactics to improve the organization as a whole.

Going forward, those whose experiences are reflected in this report sincerely hope that our organization will learn to be more elastic, more tolerant of novel, effective approaches to solving problems. Our team has always operated in the spirit of Peter Senge's *Learning Organization* [3], by using feedback loops internally to continuously cooperate, adapt, and improve.

## 5.6 How Did We Fare?

So, yes it was worth using the agile approach, but in the absence of the full confidence and backing of an executive, you must prepare for sacrifice. In the high compliance mode, the whole team may pay. In the low compliance (dare I say, defiance) mode, the project manager must pay. In the stuffer veneer approach, the stuffer must pay a modest price. There is no free lunch in an inelastic organization.

We fared well in terms of our deliverables, but not so well in terms of personal and professional sacrifices. Next time, we will seek an executive sponsor who can provide the necessary external support and recognition. We will know to prepare for the Stuffer role from the start. It was worth the experience to learn more about the organization and how people behave in a variety of circumstances. It was a long route, but an educational one, and in the words of Winston Churchill:

*“You can always count on Americans to do the right thing - after they've tried everything else.”*

## Acknowledgements

I owe much of my learning on our recent projects to the very talented, committed and sensible software engineers with whom it has been my pleasure to work on the Embedded Java Enablement Team: Paul Vanderlei, Bill Millett, Scott de Deugd, Brett Hackleman, James Branigan, Simon Archer, Patrick Dempsey, Randy Carroll, Eric Yokeley, Dave Lavin, Bruce Hyre, and Pat Huff. Their integrity, candor, and work ethic have not only made them the finest of colleagues, professionally, but also great friends, personally. The team exemplifies the saying “the whole is greater than the sum of its parts.”

## References

- [1] <http://sysdev.ucdavis.edu/WEBADM/document/radmanage-studies.htm>
- [2] Larson, Carl and Frank Lafasto, *Teamwork*, Sage Publications, 1989.
- [3] Senge, Peter, *The Fifth Discipline*, Doubleday, 1990.

## Biography

John Cunningham began practicing agile techniques long before they were named formally. Having begun his career as a consultant at Andersen Consulting (now Accenture) and Computer Sciences Corporation (CSC), he was immersed in stringent waterfall development methodologies. He received his Project Management Certification in 1991 from CSC and completed the company's methodology courses and was certified to teach them. In the early 1990's he worked closely with CSC[Index Group to define a light weight methodology for Business Process Reengineering that packaged business objectives, use cases, working software, rapid iteration, and testing in timeboxes.

He then took this toolkit with him to Travelers Insurance to put it in practice managing the delivery of Smalltalk based systems solutions. Having proven the approach to be workable, he joined Object Technology International (OTI) where he led customer facing enablement teams. After OTI was acquired by IBM in 1996, Mr. Cunningham continued this role as Engagement Lead and Project Manager for several global embedded Java engagements as a member of the Embedded Java Enablement Team (eJET).

Mr. Cunningham holds the following degrees: BS in Mechanical Engineering (Columbia University), MS in Mechanical Engineering (University of Massachusetts at Amherst), and an MBA in Finance (University of Connecticut).