Case Study at the UK Ministry of Defense

> *There can be no substitute for the clear, positive ID of targets linked to unambiguous confirmation of precise location. The passage of positional data relating to both the target and the nearest friendly forces should be mandatory.*
>
> Board of Inquiry Report,
> Ministry of Defence, 2004

We shall see proof in this chapter that an agile approach can incrementally deliver large mission and safety critical technology solutions. In this case, it did so quickly and is now on its way to protect the lives of coalition service personnel. It shows how the UK Ministry of Defense (MoD) successfully developed a new, improved battlefield system in the space of 18 months by using the DSDM framework.

In relating the case, I will give some concrete examples of the concepts behind agile.

We will see how the DSDM framework was used to provide governance and a project management approach to ensure that things got done on time and within budget. This is because it is important for you to be exposed, at least at an overview level, to some of the essential jargon that agilists use, and to get a gist of the processes they are advocating. In later chapters, I will describe the Scrum method and XP techniques which are also popular methods and are complementary to each other and the DSDM framework. As mentioned in the introduction, each of these three sets of best practice addresses development project issues at different levels. But always bear in mind that one of the arguments of this book is that although best practice materials such as these are helpful, it is the leadership of management and those inside the teams that really make projects like the one in this chapter a success.

At the end of the case study I ask some probing questions that should prompt you to refer back to the text and provoke you into thinking more deeply about how you can adopt the agile leadership lessons therein.

Case Study Background

On January 14, 2009, Captain Tom Sawyer, 26, of the Royal Artillery, and Corporal Danny Winter, 28, of the Royal Marines, tragically died in a ‘friendly fire’ incident in Helmand

province. A subsequent investigation revealed that they were killed by a heat-seeking missile fired by coalition forces in bad visibility while they were providing mortar ground support.

Many such incidents have occurred during combat operations in Afghanistan. This incident increased the number of British troops killed by friendly fire in Afghanistan operations to six. Incidents of friendly fire are usually due to a lack of situational awareness of the combatants, not due to a lack of precision in the weaponry. Responsibility for command and control of fire is dispersed to individual units in the heat of battle, and the knowledge of who friendly units are and where they are situated is vital to those responsible for fire control in modern, fast-moving battlefield situations.

Poor situational awareness in combat is a key risk factor, often leading to friendly fire deaths. The board of inquiry into the killing of Lance Corporal Matthew Hull in Iraq in 2003 found that the co-ordination between battlefield units and air units was lacking due to poor situational awareness.

The CIDS Project

The US, UK and other NATO forces have been developing and improving Combat Identification Systems (CIDS) over many years. In 2009 the UK Ministry of Defense (MoD) initiated a project to create a Combat Identification Server (CIDS). The CIDS was needed to tightly integrate close air support with shared situational position information.

A contract was awarded to General Dynamics to develop the CIDS to be in place by July 2010.1 It needed to provide autonomous, accurate near real-time force tracking and location information to direct fire away from coalition troops. General Dynamics had only 18 months to integrate their “Net-Link tactical gateway” with specialist technology supplied by its subcontractors, Rockwell Collins and QinetiQ. Every few seconds, CIDS would integrate data from all the friendly forces in a battlefield and distribute it back to all the nearby unit commanders.2

Project Kick-Off and the Foundations Phase

To meet their objective of an 18-month implementation of this lifesaving software, the MoD chose an agile approach. They believed that complex military technologies could be better delivered without delay or unexpected cost overruns using agile.

A decision was made to use the DSDM framework because it gives guidance on the process for agile supplier delivery to a customer. The customer does not need to be a third party – it could be the technology department within an organization. The important point with DSDM is that it focuses on what the output will be, rather than how it will be developed – a product centric approach. Therefore, it can be used to formalize payment milestones with suppliers based on product deliveries.

The first phase of the DSDM development was the Foundations phase. On the CIDS project the team analyzed the theoretical payment plan in the contract during their Foundations phase and found that it did not match reality. Both the MoD and General Dynamics recognized that a win/win situation was needed, and that traditional contract

renegotiation would take time, and could lead to a deterioration of relationships before the development had even begun. They agreed to start work, and use evidence of progress to amend the scope of the required solution to fit with the planned timescales.3

How DSDM Avoided the Pit-Falls of Waterfall Projects

DSDM requires just enough design up-front (EDUF), and the Foundations phase should be as short as possible, while still ensuring an essential understanding and clarity of structure of the overall solution and to create an Agile plan for delivery. This initial Foundation phase created an architecture that gave both the MoD and General Dynamics an assurance that minimum acceptable performance levels could be achieved. 4 For example, tracking information on the position of friendly forces needed to be collated from a minimum of 15 different units in any battlefield. The architecture also had to be flexible enough to provide real-time position information to not only artillery units, but also to nearby aircraft. 5

The approach ensured that test and evaluation of the solution was a “constant and regular activity”, and allowed the development team and the stakeholders to gain more confidence with each iteration.

The project would run from February 2009 to July 2010. Overall plans were agreed at the end of the Foundations phase, which took three months. Then the Exploration and Engineering phase started with three iterations, each about 3–6 months long:

- Iteration 1: Create a simple version of the software that could deal with one friendly force position
- Iteration 2: Extend the software to process multiple position information
- Iteration 3: Make the solution robust and fast enough to deal with the operational number of request responses and to interface with systems from other coalition partners.[6]

The MOD planned practical demonstrations for June 2010, before final deployment took place in July 2010.

DSDM stresses the need for scalability from the smallest project to the very largest. It concentrates on governance and structures around incremental project outputs. It was first published in the UK in 1994 as an alternative rapid development method, which would avoid the pitfalls of the traditional waterfall approach.

Waterfall projects are segmented into discrete phases, each dependent on the completion of the previous phase, but without feedback or iteration. When using a waterfall approach, one cannot start a phase until the previous has been completed. This leads to a series of one-way ‘Gates’ (see Figure 1). Once one has committed to swimming downstream, it is impossible to return to an earlier stage without a lot of effort – similarly difficult to attempting to swim up a waterfall. In contrast to doing just enough design, a waterfall approach requires a grand design in detail before any solution building commences. A waterfall approach is appropriate for some civil engineering projects that are monolithic in nature, such as building a skyscraper, but in technology projects a waterfall approach will tend towards what Kent

Beck called ‘Big Design Up-Front’ (BDUF) when describing a fundamental problem of the waterfall life cycle – that it relies upon pinpoint accuracy and perfect logic at every step if it is to produce a workable solution.7 Kent’s argument, and one that I emphasize in this book, is that we should aim for Enough Design Up-Front (EDUF), not BDUF.

Although DSDM started as a proprietary method closely controlled by a small consortium, in 2007 the decision was taken to make the method more openly available. The manual is now available to all for free on the Internet at www.dsdm.org and training may be bought from many suppliers (subject to training body certification requirements of the DSDM consortium). 8

At 202 pages, the DSDM handbook may not at first glance appear to reflect the ideal of a ‘light-weight’ method. However, it supplies the role and process definitions often required for large projects by government regulations, and thus provides a useful template for a project management framework. It is process and output orientated, and gives seven main steps for every DSDM project, creating 43 products – each described in the handbook in some detail. Prior to the Foundations step, the customer (perhaps with the help of expert suppliers) should carry out the Feasibility step.

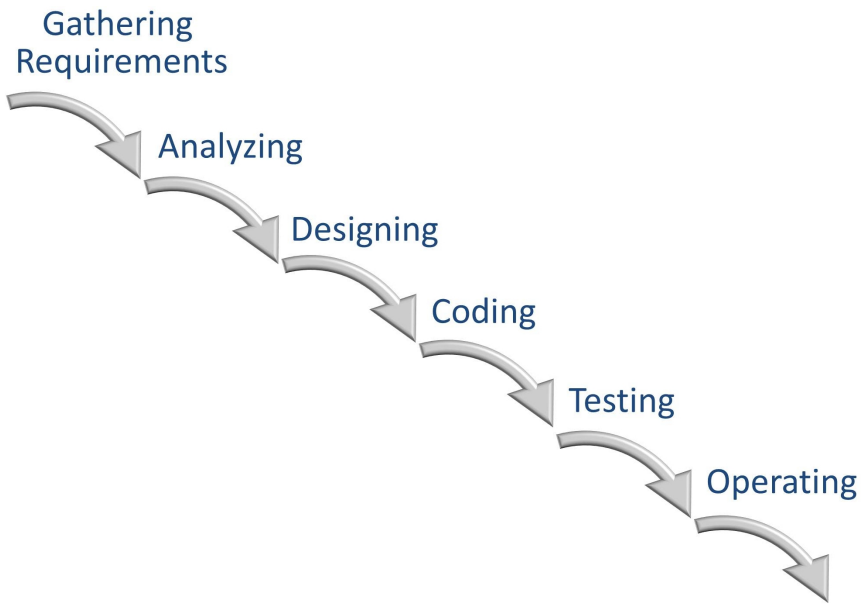

Figure 1: An example of a waterfall life cycle

The method gives guidance as to the level and approach needed to produce an outline business case containing enough information to make a decision, but no more. If the project is given the go-ahead, then in the Foundations phase of the project this business case is expanded just enough for internal needs and government regulations. After the project is finished, DSDM gives advice on collecting lessons learned, evaluating the project performance against expectations, and monitoring the business performance of the solution against the business case.

One of the strengths and flexibilities of DSDM is that it gives guidance on how the iterative development work of Exploration and Engineering should be carried out alongside Deployment. It encourages flexibility in how these could be combined together, or omitted.

Some projects initially need to iterate Exploration and Engineering many times, building models of different solution options, before proceeding with the iterative Engineering of a solution and its deployment. For example, if one-month iterations are being followed, but updates to end-users are restricted by a wider organizational policy to once every three months, then only every third iteration will include a Deployment step.

Iteration and feedback is the core of DSDM, and it makes it very different from waterfall approaches. Its strength is that it presents agile concepts from a management point of view, using terms that traditional project managers understand while avoiding a waterfall approach. Like many methods, though, it has little to say about leadership behaviors. Processes and outputs are defined that are amenable to 'traditional' project management techniques, but have an agile approach. For example:

- Quality planning is used to define the necessary levels of acceptance for project outputs – this provides a description for each output that can be objectively tested and audited (see *definition of done* later)
- Requirements planning is used to maintain a Prioritized Requirements List (PRL), with mandatory release dates defined for all mandatory requirements, and tentative release dates for others
- Earned Value Analysis (EVA) can be carried out to compare the actual versus estimated development effort originally expected for each product feature in the PRL, thus providing feedback on the accuracy of the original estimates and the productivity of the team. (EVA is a technique that is controversial with agilists, as discussed further in Part III.)

Thus a high level of compatibility with traditional formal management techniques can be achieved, but coming from the direction of flexibility and iteration, rather than upfront, detailed plans that become set in stone as baselines to be measured against.

The DSDM framework is an agile approach and guards against cost and time overruns by turning the baselining model on its head. In a waterfall project, a detailed baseline for the scope of a project needs to be agreed upon – supported by detailed design assumptions and theoretical estimates. Hence the phrase Big Design Up-Front (BDUF). If the estimates are inaccurate (and of course they often are because they are made before work begins and actual progress starts to be measured) the only variables left in the equation are cost and/or timescales.

Stakeholders flex their muscles and ask for additional nice to have features which cause the required amount of work to increase: a situation known as scope creep. This is why so many waterfall projects go over time and cost. Since the baseline is fixed, these mutually dependent parameters are allowed to run out of control. And what is more, waterfall projects usually implement one risky, disruptive, large change to operations: the all at once or big-bang approach which we will encounter again and again in the stories of large project failure in this book.

DSDM is an agile method and therefore has a different philosophy from the waterfall approach. When it is used as the project management framework to guide the team, only the central core of solution features is identified at the outset. The scope is allowed to change, in

a controlled manner, as the inevitable mis-estimation of time and cost becomes clear. The opposite of scope creep takes place – scope is reduced if difficulties are encountered, rather than time and budget being increased. The project comes in on time and cost because DSDM fixes these variables, and instead re-scopes the features to be delivered. In effect, there is zero time or cost contingency, but there is contingency in the scope of requirements (see FIGURE 2).

At its simplest, features left out of one iteration are simply deferred to the next iteration. This can work both ways: if better than expected progress is made, then features that were only on a wish list for an iteration may be included – some delight and surprise for the stakeholders!

DSDM suggests that no more than 60% of the work expected for each iteration of development should be on features classified as Must Haves. About 40% of the remaining work is split between Should Haves and Could Haves. The Should Haves are features that would be painful to leave out, but a workaround could be found for them otherwise a Must Have would be compromised.9 Could Haves are features that bring additional value-add and business benefits, but can be delayed for future work without any immediate downside. To complete the picture, and to ensure that limitations to scope are understood, some requirements are classified as Won't Haves.

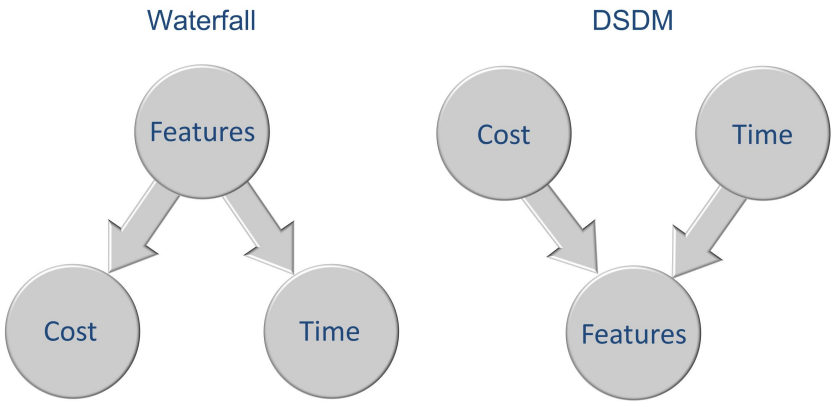

Figure 2: Waterfall: Features are the driver — DSDM: Cost and time are the drivers

Of course, 60% is a rough rule of thumb. As the project progresses, the team's velocity will be calibrated against the PRL. Each PRL item can be sized using the idea of story points rather than notional person-days. These story points are a relative measure of the size of each item. This concept cuts away the idea that plans can be accurately estimated in detail up-front. Progress is measured as the number of story points per day per team member, not the number of person-days notionally assigned to a set of detailed tasks at the start of a project.

It is only when the team gets going that the actual rate of progress of that particular set of people, technology and problem domain can be determined – by feedback from actual experience, rather than conjecture and theory.

The actual percentages should be reviewed with regard to the predictability of the overall scope of the project and the calibration of the velocity of the team. If the scope is well understood, in a stable business environment, and the target technology has been previously

used, then perhaps a lower percentage of requirements could be in the tentative category of 'Could Have's'. However, it is tempting to make simplifying assumptions and start to move back towards traditional fixed-scope estimating. The risk is that the assumptions may be false, and development will then be more problematical than expected. It is better to achieve an over-delivery of output features than promise too many mandatory features and not deliver.

Requirements Planning in DSDM

A key control in DSDM is the list of requirements or Prioritized Requirements List (PRL). It lists all the requirements and states which are most needed for the upcoming iteration.
Every requirement is prioritized into one of four categories, referred to by the acronym 'MSCW'. Often this is pronounced and written as MoSCoW. These are the Must Have, Should Have, Could Have and Won't Have requirements. This technique, which is central to DSDM, helps create flexibility and Agileness by three tricks:

- Priorities are set within the framework on iterations which are *timeboxed* – the deadlines are immovable. The team has delegated responsibility as to which *Should Have* and *Could Have* features they will deliver. Of course at the core of their work are the *Must Haves*. When a delivery is to be deployed into live use, rather than as a prototype demonstration of capability, the *business sponsor* (who is the executive responsible for the success of the project) and the team members (including key users) should create a *joint deployment plan*. Decisions are set at the lowest level possible so as to reduce the cycle time in decision-making and ensure that quality and delivery timescales are met.

- Priorities are set for each iteration and change as the project progresses. For example, features that are *Should Haves* for one iteration may be promoted to *Must Haves* for the next iteration.

- Quality is protected: if an essential feature in the emerging solution is not of sufficient quality (it functions incorrectly, is unusable, or cannot meet capacity or other performance requirements) then it can be descoped from delivery for that iteration. Mike Cohn notes that the *Could Have* requirements items in an DSDM Prioritized Requirements List work as a *feature buffer* which can be sacrificed as required so as to ensure deadlines are met.[10]

DSDM ensures that the necessary governance is in place so that if any Must Haves are likely to fail these tests, a business sponsor is in place and responsible for decision-making. In these cases it is usually necessary to change the deployment plan. Effort is always focused on ensuring that the highest priority features are of adequate quality. Research indicates that on average only 45% of features of technical solutions are used to any great extent, so a ruthless approach to descoping Could Have requirements is needed if the overall project is to

produce early benefits and have a robust business case.11

A Solution to 'Friendly fire'

Forward Air Controller engagement scenarios and acceptance criteria were developed with real-life military operators collaborating in the Exploration and Engineering iterations. These tests were indexed against the requirements itemized on the PRL.
For example, air requirements such as interfaces to and from the "Link 16" air intelligence system were Must Have requirements, whereas armored situational awareness systems, such as the Force XXI Battle Command Brigade and Below (FBCB2) system, was categorized as being a Should Have requirement.12

Developing the CIDS in Timeboxes Within Each Increment

Each of the 3–6 month long increments was divided into timeboxes of a month. In each timebox, the team, which included domain experts, was encouraged to get on with the work without interruption. Deadlines were sacrosanct. If within a timebox it became apparent that any of the Must Have requirements were at risk, then the team self-organized the redeployment of people away from development of solutions for lower priority requirements. This is the essence of Agility: decisions are delegated to the lowest possible level – to those closest to the work at hand.
A key discipline of Agile is that deadlines cannot be extended. Any issues become evident immediately and a positive attitude towards failed tests is encouraged. It is better to fail early and rectify an error, than to hope for the best and carry on regardless. In this case, through a requirement trading process the MoD agreed that a few Should Have and Could Have requirements could be descoped from the PRL. The important factor was that agreement was achieved without any penalties being incurred on the supplier, or any cost or schedule overrun for the customer.13
The supplier project manager from General Dynamics led a multi-disciplinary team comprising staff members from the other sub-contractors (Rockwell Collins and QinetiQ) and also MoD staff and their specialist technical advisors. The MoD designated an overall business sponsor (responsible for the success of the whole project), and a business visionary (responsible for decisions on day-to-day issues). Risks were recorded on a risk register and linked to items on the PRL to provide a means for prioritization and replanning. Again, the overriding concern was to maintain an iron grip on cost by flexing the delivery of functions to deal with risks as they emerged.
On this project the potential for 'tit-for-tat' negotiations over points of detail and costing were considerable. Trust had to be built up between a potentially suspicious customer not used to an agile approach, and a supplier under pressure to deliver.
The MoD procurement division initially proposed a severe penalty clause to guard against the possibility that the supplier would not deliver all the requirements – even the Could Haves. However, a collaborative approach was agreed upon, following the DSDM principles of fixed cost rather than fixed scope. Any difficulties encountered were to be resolved by requirement

trading. In effect the project contingency was held in the Could Have requirements which could be traded out if unworkable or too onerous.14

Laboratory Integration Tests

As planned, at the end of summer 2009, integration tests started to take place at the UK Battlespace Laboratory. The Battlespace Laboratory is an independent body managed and owned by the MoD. It brings together government, industry, and military coalition partners from across the world to collaborate on highly realistic simulations of battlefield conditions.15
A test took place at the Battlespace Laboratory at the end of every 3–6 months in line with the development team delivery iteration schedule. The aim was to work towards a final defense demonstration servicing 50 interoperating battlefield positions. This testing was at Technology Readiness Level 6 or TRL6 (explained in more detail later on page **Error! Bookmark not defined.**) and had to be seamless and free of any significant bugs. A realistic demonstration to the military was carried out at the end of each laboratory test to increase their confidence.
Testing was based on a number of scenarios. In agile teams each of these is called a user story, but given the special context of the battlefield the team used the term vignette. One vignette, for example, was based on a land battle group carrying out counter-insurgency operations. The simulated mission was for the UK to coordinate a multi-nationality NATO attack on an insurgent compound in a desert storm. Proof was needed that the system would be able to provide friendly force ID to all units (including artillery and attack aircraft) in a difficult environment.
The team found that an agile approach instilled a discipline of delivery into the formal test environment at the end of every iteration come what may. An agile approach of immoveable deadlines ensured that intensive use could be made of the Battlefield Laboratory on the expected dates, thus making best use of an expensive and limited facility. A focus on interoperability was the key to the development. Although various items were re-prioritized for each iteration, in the end, the flexibility and discipline of the DSDM framework adopted meant that important requirements were not sacrificed and CIDS and the other battlefield systems all linked up to each other successfully.16

Joint US and UK Interoperability Testing

Full coalition interoperability testing with all coalition partners at TRL8 (the highest technology readiness level) was to take place the next year at the next “Bold Quest” coalition Combat ID (CID) capability assessment organized by the Joint Forces Command for 2011. These demanding exercises are aimed to enhance situational awareness, targeting, and minimize “collateral damage and fratricide”.
Rather than wait for that event, the MoD decided to carry out a previously unscheduled trial to prove the system at TRL 7 well in advance of “Bold Quest”.
So, in Norway, in August 2010, the new UK CIDS system was demonstrated side-by-side with the US CIDS by joint coalition ground and air forces – and all were able to successfully

communicate with one another.17 Both static tests and dynamic tests were undertaken using known positions and mounted and dismounted Norwegian soldiers exercising controlled scenarios.

Over 90 dynamic user friendly force information requests were made from the ground and the air using seven different systems.18 Throughout the exercise period, the CIDS proved to be highly reliable, providing friendly-force position data within an average of three seconds to within five meter accuracy.

In addition, even though the Danish aircraft had not been specifically prepared for the exercise when they arrived, they were immediately able to make successful requests for friendly force ID using their Link 16 technology.19

Conclusions

In this first of five case studies we have seen how an agile approach delivered a large mission, and safety critical technology solution, and we explored some of the concepts behind agile, and the DSDM framework.

Questions

1. The MoD had indicated their inexperience with agile approaches. What risks did this represent to their business case?

2. What strengths and weaknesses are there in the application of DSDM to the CCID project outlined above?

3. The MoD procurement division was keen to ‘nail down’ the suppliers to a fixed specification. What may have been their thinking? How would you draw up an agile contract that would fairly hold a supplier to account for poor performance? How would it also ensure that the customer is held to its responsibilities.

4. The Agile Manifesto Principles expect projects to iteratively deliver working solutions and have a natural preference for shorter rather than longer timescales between iterations (see Table 2). Did the CIDS project meet these criteria? Could more have been done to make the project more agile?

Look at the Henson’s presentation of the CIDS project plan (see Endnote 20). Compare it with the waterfall life cycle (see Figure 1). What similarities are there? What differences?

[1] Williams, Hodgson 2009

[2] {General Dynamics UK 2011 #357}

[3] {General Dynamics UK 2011 #357: 5}

[4] {Craddock 2012 #330: 2}

[5] {General Dynamics 2009 #352}

[6] {Henson 2009 #348: Slide 7}

[7] {Beck 1999 #62}

[8] {DSDM Consortium 2008 #165}

[9] Thanks to Dot Tudor for pointing out to me that if workaround is possible, but painful, then a requirement should be considered a *Should Have* rather than a *Could Have.*

[10] {Cohn 2005 #1: 189}

[11] {Proceedings of the 31st EUROMICRO 2005 #392: 6}

[12] {Henson 2009 #348: Slide 9}

[13] {General Dynamics UK 2011 #357: 4-5}

[14] {General Dynamics UK 2011 #357: 5}

[15] {Madahar 18/06/2012 #353}

[16] {Henson 2009 #348: Slide 13-14}

[17] {New combat ID tool would #354} and also {saarelainen 2011 #355: 136}

[18] MIDS-Link 16, BOWMAN, Ground Asset Tracking System,, Variable Message Format, FAC NAV, NORTAC and NORMANS

[19] {3SDL 2010 #356}

[20] {Henson 2009 #348: 7}