Dynamic System Development Method

Submitted by[*]*:*
Benjamin J. J. Voigt
Zürich, Switzerland
S99-728-123

Supervisor:
Prof. Dr. M. Glinz

Advisor:
Dipl.-Inf. C. Seybold

Department of Information Technology
University of Zurich

Zurich, 20 January 2004

[*] I would like to thank Gregor J. Rothfuss for his review.

Table of Contents

1. Introduction

Methodologies in today's professional work environment are means of guiding people, resources and processes in a defined and repeatable manner. The more people are researching methods, the more clearly common patterns emerge which are becoming a guide to develop new methodologies. The Dynamic Systems Development Method is a framework which embodies much of the current knowledge about project management. DSDM is rooted in the software development community, but the convergence of software development, process engineering and hence business development projects has changed the DSDM framework to become a general framework for complex problem solving tasks.
The aim on this paper is to describe the DSDM framework and show how it relates to the agile movement principles. In the end a short overview of the success of DSDM and relates to similar development methods like SCRUM or Crystal in the conclusion. [SUTH03], [COCK02]
The reference material for this paper is the book by Jennifer Stapleton DSDM Business Focused Development.

2. Values and Principles

The DSDM framework can be implemented for agile and traditional development processes. To show how DSDM relates to the agile methodology it's essential to understand how DSDM principles relate to agile development process values. The following chapter introduces the DSDM principles and matches them to key agile concepts afterwards. In the last two sections a brief analysis of the DSDM philosophy will conclude with a reflection on whether DSDM introduces any new concepts.

2.1 The 9 Principles

The following nine principles are essential to any DSDM implementation, ignoring one of them will break with the frameworks philosophy and significantly increase project risks. This is in sharp contrast to the DSDM project structure, where some of the steps may be disregarded or modified to fit a concrete project layout.

2.1.1 Active user Involvement is Imperative

The first principle is considered the most important, because user involvement through out the project effectively reduces errors in terms of user perception, and therefore reduces error costs. Instead of working with a large set of users a DSDM project guidelines recommend working with a small, select set of users continually, rather then in periodic workshops or review sessions. Continuity is required by several other DSDM principles as well, highlighting this single project attribute even more.

2.1.2 Teams Must be Empowered to Make Decisions

To proceed as quickly as possible transaction costs, as resulting from friction in communications of project participants and managers, need to be avoided. Requesting authorization of even modest resources, or simple requirement changes will slow down a project significantly. Addressing these inefficiencies users and other DSDM participants should be given limited authority to make decisions related to:

- Requirements in practice
- Which functionality needs to be in a given increment
- Prioritization of requirements and features
- Fine details of the technical solution

2.1.3 Focus on Frequent Delivery

Frequent deliveries of results ensure that errors are detected quickly, are easily reversed and closer at the source of the error. This applies both to program code as well as to documents like requirements or data models.

2.1.4 Fitness for Business is Criterion for Accepted Deliverables

As the name of the DSDM framework suggest, its primary endeavor is to deliver software which is good enough to solve the business need and bother with any

enhancements in a later iteration. Refactoring, design engineering and feature enhancement, being part of the natural live cycle of any software system, need to be recognized as integral part of the project, rather then being a task which is taking place only after the project is finished. DSDM does not promote to write ad-hoc software, but suggest satisfying the business needs first, and acquire timeboxes for refactoring and related activities in a later iteration. Even on a DSDM project it's crucial to identify the key issues, which do required a robust design. Agile methods in fact do require a sound understanding of patterns and architecture; depending much more on patterns then traditional development methods, because it is fare more likely to make a decision which will turn out to prohibit delivering iteration results three or four iterations into the future.

2.1.5 Iterative and Incremental Development is Mandatory

In order to keep the complexity of the project manageable, it needs to be decomposed into small feature packages; with each release adding new features until the complete set of business requirements are fulfilled. This principle requires accepting the fact that any software system is subject to change. This principle can easily be introduced even at the beginning of a project, since specifications and other results can be produced in an iterative manner as well. The smaller the increments, the easier a response to changed requirements is possible.

2.1.6 All Changes During Development Must Be Reversible

Being responsive to change requires that system configurations are changing during the development of any one increment due to changed priorities in the requirements. Modern software tools support a dynamic configuration of projects as required by this principle. It's often feared that reversing a development process will result in loosing precious previous work, but since DSDM advises to iterate though small increments, the total loss of work is very limited.

2.1.7 Requirements are Baselined at High-Level

To limit the degree of freedom to which requirements can be altered during the development process, some high-level requirements need to be established. This baseline which is to be interpreted as a requirements "freeze" is agreed upon during the business study phase of the process.

2.1.8 Testing is Integrated Throughout the Lifecycle

Many development methods ask for testing as late as the design or implementation phase. DSDM requires testing early in the development process. Even testing interview documents by cross checking them with a control group, or similar techniques.

2.1.9 Collaborative and Co-operative Approach

Avoiding separation and encouraging collaboration of technical staff and business staff in a project is mandatory during DSDM projects, because co-operation is crucial to succeed in a DSDM project. Without an atmosphere of trust and honesty it will be

hard to gather requirements, and later getting honest feedback on the resulting products.

2.2 Values

The Agile Manifesto claims 4 values and 12 principles, and is considered the beginning of agile development methods. To show how DSDM relates to the agile philosophy the following sections will associate the DSDM principles with the value system as described the Agile Manifesto.

2.2.1 Individuals

Many modern management practices highlight the importance of humans, so does the Agile Manifesto and DSDM framework. The DSDM principle 1, 2 and to a lesser extend 9 highlight the key role individuals take in a project. Since DSDM projects need to implement all of the 9 DSDM principles it's implicit to satisfy the agile demand on rating individuals over tools and processes, which are, however, still required by the DSDM framework. The high degree of interaction DSDM requires are less obviously hidden in the principles 5 and 3, both of them demanding a healthy work environment, where communication and coordination can take place as friction less as possible.

2.2.2 Working Software

Placing the business needs into a prominent position, DSDM principle 4 commands the success of a project to be measured against the business value; technical aspects are of no concern to the users. This is a highly controversial value, more often leading to an ad-hoc solution being justified with this argument. First it's important to keep in mind that the agile values express mere preference orderings when time constraints limit the implementation of project management methods, and secondly working software includes working in any future environment. It's fine to hack a quick solution, when the next increment won't contain any new features and developers are allowed to fixes the design. These decisions need to be reached in cooperation with the users involved in the project. Principle 7 is crucial to ensure that a common understanding of the definition of "Working Software" exists, the baseline abstracts development goals, which are fixed, and cannot be changed for a given development target.

2.2.3 Collaboration

As described in the two previous sections, collaboration is most significant to DSDM (principle 9) and agile methods likewise, because these development paradigms react to change and whenever change is happening, regardless of being part of a project or in personal life, a fair amount of communication is required sorting out the implication of change. Being able to talk without political barriers helps a lot; the need to interpret other people's statements and the error of misunderstanding is greatly reduced. Unfortunately collaboration is not easy, since humans tend to take selfish decisions; especially if an immediate reward is expected. Creating an

environment where collaboration works as seamlessly as possible is a management and leadership task.

2.2.4 Responding

Welcoming change and responding to change is rooted in principle 6 and partially in principle 7. The management of changing business demands is commonly perceived as the most prominent difference of agile and traditional methods. DSDM handles change in a variety of ways, mainly through prioritization as described in Chapter 3.

2.3 The DSDM Philosophy

The DSDM framework itself is a very dynamic and modular system. While developing, DSDM designers were interested in the "edge case" rather then a mere project "ingredient", meaning that it's most important to know what choice to make when time constraints are applied.
DSDM doesn't require it's user to implement the whole project structure, it only demands strict obedience to the 9 principles, apart from that, any project manager can implement a development process resulting to be more or less agile, depending on the situation and the constraints, even combining DSDM with other methodologies is allowed, even welcome, in several environments. The core idea of DSDM being: "No plan will ever survive the first day of development." is why the framework recommends many techniques to handle these dynamics. Finally DSDM does not aim to solve all development problems, when the 9 principles can not be implemented, DSDM is most likely not the best choice.

2.4 Yet another Guide on Good Project Management?

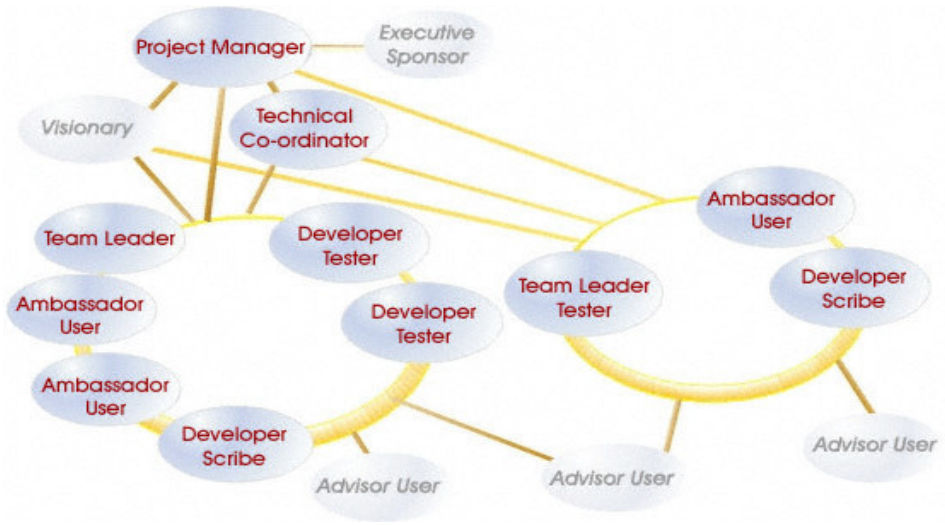
The DSDM principles suggest that a lot of the current best practices where collected, synchronized and consistently wrapped into a single methodology. That's exactly what happened, and what the intentions were in 1994 when the first version of DSDM has been released. Nowadays version 4.2 has been released. DSDM will always continue to be a best practice framework, but rather then other best practice frameworks the DSDM consortium acknowledges that the environments of many projects change, and adapts the framework to these requirements. [DSDM04]

3. The Project Structure

The DSDM project consists of 7 phased steps which are organized and embedded in a rich set of roles and responsibilities and are supported by several core techniques. This chapter will highlight the project logistics in the first two sections and describe the process afterwards.

3.1 Roles and Responsibilities

Several role classifications exist within a DSDM project, namely project, team and workshop roles. Every person takes one of the roles, no distinction between the IT staff roles are made; regardless if a person is programmer, designer or analyst, they are all referred to as developer, one exception being the tester. Another group of people in a project are users referred to as ambassador users if they are fulltime participants and advisor user if they are only part-time members of the team.
Further fulltime roles on a team are the team leader, project manager and the technical coordinator. In single team projects team and project leader might be represented by the same person, but in projects with more then one team (possibly a separate test team, or a team working on different tasks) the project manager takes coordination responsibilities and the team leader takes specific technical responsibilities related to his teams tasks. In any event the project manager represents the link between the user and the IT community. Coordination of technical aspect like architectural and quality issues in a multi team project is the responsibility of the technical coordinator.
To secure motivation and sufficient resource support every DSDM project should also include a visionary user and an executive sponsor as part-time participants. The visionary is committed to motivate the team, and align the projects goals with the business goals, where as the executive has to commit resources to the project as well as establish the proper relations with other executives related to the project.



Relationships of Project Roles - Figure 1

[DSDM04a]

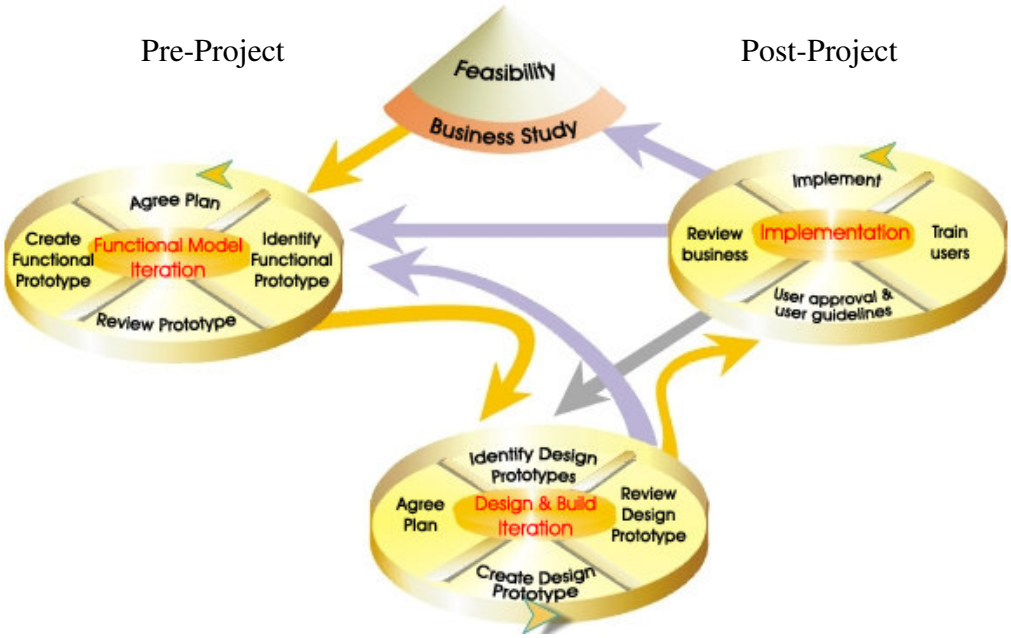
3.2 Team Organization and Size

The usual DSDM project consists of one or two teams, where any second team might take the responsibility to run tests on the development team products. Organizational research suggests that more than 5 people (excluding external experts) on a single team are ideal; it is strongly advised against larger then 6 person teams. If more work has to be done than a single team can deliver, a multi team structure is required, DSDM projects have documented project sizes of up to 150 people. In large projects the decomposition of tasks may not only be functional, but should be task oriented as well. Several projects found that a testing team and a development team for any single feature set are useful, the rational being, that the larger the project, the most important the quality of each component is. Since testing is a destructive labor, it's usually more effective not to leave it to the author of a product. Teams are assigned to a task/timebox bundles (see chapter 4 for details on timeboxing). Figure 1 illustrates a simple team organization where the test and development tasks are decomposed into different teams, coordinated by technical lead and project manager. All gray symbols related to part-time participants. [YEBO83]

3.3 7 Phases to Rule Them

The DSDM development process consists of 7 phases, of which some might be omitted in a concrete project implementation. Each phase owns several key tasks (see figure 2) and can be modified to include more tasks as appropriate, which might be required when combining DSDM with other development methods. Especially DSDM does not specify which technical tasks are to be carried out in each iteration, allowing DSDM to be used in a wide variety of situations and projects, because it allows to populate it's framework by individual organizational practices. Three phases are designed as iterations, meaning, that they are to be executed in each increment.

DSDM does specify concrete results for each task and for each one of the three phase groups (FMI, DBI and Implementation). However the result specification is general enough to use DSDM in engineering projects and business projects alike. The following describes the 7 phases as shown in figure 2.



Project Phases - Figure 2

[DSDM04a]

3.3.1 Pre-Project:

The Pre-Project phase includes project suggestion and selection of a proposed project candidate. The pre-project determines if a project should be realized at all.

3.3.2 Feasibility Study

The normal considerations in a Feasibility Study are a definition of the problem to be addressed, assessments of the likely costs and technical feasibility of delivering a computer system to solve the business problem.

3.3.3 Business Study

Having decided in the Feasibility Study that DSDM is an appropriate method framework, the Business Study provides the basis for all subsequent work. Like the Feasibility Study, it is as short as possible (the duration measured in weeks rather than months), while achieving sufficient understanding und the requirements.

3.3.4 Functional Model Iteration (FMI)

The focus of Functional Model Iteration is on refining the business-based aspects of the computer system, i.e. building on the high-level processing and information requirements identified during the Business Study. A given project may have several FMI type timeboxes.

3.3.5 Design & Build Iteration (DBI)

The Design and Build Iteration is where the computer system is engineered to a sufficiently high standard to be safely placed in the hands of the users. A given project may have several DBI type timeboxes.

3.3.6 Implementation

The Implementation phase covers the cutover from the development environment to the operational environment.

3.3.7 Post-Project

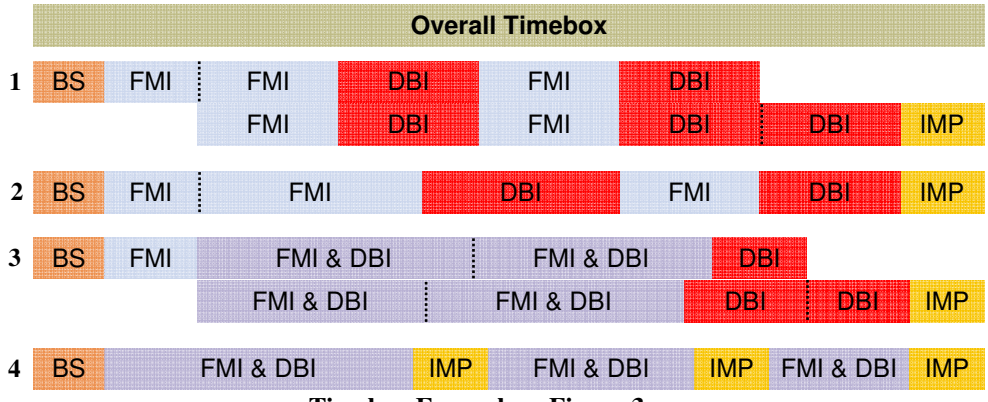
Post-project tasks include measurements on how the deployed system is performing and if any further enhancements are required. Usually these measures take place about 6 month after the project technically finished.

4. Recommended Core Techniques

4.1 Timeboxing

Traditional project management uses milestones to agree on a deliverable to a given point in time. While milestones work well enough, a timebox is a much more powerful tool to achieve the same result. A timebox is an interval, usually no longer then 6 weeks, where a given set of tasks should be achieved. The reason for the relatively low duration of timeboxes is the fact, that humans give much more accurate estimates in the near-future involving a small set of tasks, while estimates into the distant future involving large sets of tasks turn out to have hefty errors. Timeboxes can contain several tasks, and at the end need to deliver a product. Milestones also suffer from having a fixed deliverable, while timeboxes are subject to change, since the tasks are defined, not the necessarily the deliverable, which can change if prioritization shifts during the timebox iteration, allowing for rapid response to business needs. In short: DSDM rather drops functionality in favor of delivering in time.
Looking at the four timebox examples in figure 3 we can observe several important attributes of timeboxing in combination with the DSDM project phases:

1. Timeboxes can be of different length (example 2,3 and 4)
2. Several of the same DSDM phases can be executed right after a timebox of the same phase has finished (example 1,2 and 3)
3. Parallel timeboxes are possible, even complementary timeboxes are allowed (example 1 and 3)
4. Different DSDM phases can be realized in one timebox at the same time (example 1 and 3)
5. Timeboxes can be nested (examples 1, 2, 3 and 4 are nested into the "Overall Timebox")



Timebox Examples - Figure 3

[DSDM04a]

Finally timeboxes are not the cure to all time slips, in some cases the management may decide that the cost of exceeding time is smaller then dropping some features, this is possibly the case when a new “must have” feature emerges.

4.2 MoSCoW Rules

Prioritization has been mentioned several times before. Since DSDM projects are concerned to be in time and on budget, and since the users are heavily involved into the development process, it’s mandatory to keep a constant watch on which features the user needs most. There are two reasons a user might change his opinion during the process, either he becomes aware of new technical possibilities or his work environment changes. In both cases a quick re-evaluation of the priorities is required, quick meaning, easy and straight forward classification of features into groups of different importance. The DSDM techniques to weight importance are the MoSCoW rules:

4.2.1 Must have

All features which are classified in this group must be implemented. These features are a show stopper if they are not delivered, the system would simply not work.

4.2.2 Should have

Features of this priority are important to the system and contribute a significant value, but can be omitted if time constraints endanger the delivery of any “Should have” feature.

4.2.3 Could have

These features enhance the system with functional items, which can easily be re-assigned to a later timebox.

4.2.4 Want to have

Usually these features only serve a limited group of users and are of little value.

This classification system serves as single source of decision on what to implement during the project and timebox iteration, it intentionally does not include any wish list, because whish lists without classification, and hence without a price are no good indicator for developers what users need most from their product.

4.3 Prototyping

Endorsing evolutionary prototyping DSDM projects satisfy two of the DSDM principles, frequent delivery and incremental development. Prototypes are to implement critical functionality fist to discover difficulties early in the development process, they also allow having very early deliverables to get user feedback. If an evolutionary prototyping is executed, the incremental nature allows for enhancing a

given product always considering the relevant user feedback. The necessary feedback-loop is provided by a workshop, which is the last important technique in a DSDM project. DSDM differentiates on the following for types of prototypes:

- Business Prototype: Allow assessment of the evolving system
- Usability Prototype: Check the user interface
- Performance / capacity Prototype: Ensure solution will deliver performance or handle volume
- Capability/Technique Prototype: Evaluate possible options

4.4 Facilitated Workshop

The idea of workshops is implemented in many development methods, and provides a proven tool to establish user-developer collaboration. However, several problems emerge in large teams and heterogeneous groups; the worst case is a lock-down due to too many participants or due to a knowledge gaps. DSDM tries to solve this problem by carefully selecting the right people to participate in the workshop and to leave the moderation to a neutral 3rd party if politically sensitive issues are to be worked on (i.e. decisions on operating systems and middle ware). Except for the pre and post-project phase there are 5 common types of recommended workshops (matched to the phase the are usually required at in figure 5):

1. IS requirement definition (Information)
2. Business information benefits (Business)
3. Technical system operation (Technical)
4. Acceptance test planning (Acceptance)
5. IS design

DSDM	FS	BS	FMI	DBI	Implementation
Type of facilitated worksh op	Information	Information	Information	IS design	None
	Business	Business	Acceptance	Technical	
	Technical				

Facilitated Workshop Overview - Figure 4

[DSDM04a]

6. Key Success Factors

To determine the important factors management needs to control prior to starting and during DSDM practice is crucial to the projects success. The DSDM consortium has compiled 10 most important factors from its member experiences:

1 Acceptance of the DSDM philosophy before starting work
2 The decision making powers of users and developers inside the development team.
3 The commitment of senior user management to provide significant end-user involvement.
4 Incremental delivery.
5 Easy access by developers to end-users.
6 The stability of the team.
7 The development team skills
8 The size of the development team.
9 A supportive commercial relationship.
10 The development technology

[DSDM04]

While the list leaves little to comment on, it's noteworthy that many factors are executive or even leadership responsibilities. This fact suggests that DSDM requires a fairly advanced corporate culture to be successful in the long run. Some success factors might not even apply to many projects, i.e. "supportive commercial relationships" are not usually present in Open Source Projects.

7. Related Methods and Quality

7.1 Capability Maturity Model and ISO 9001

While DSDM doesn't guarantee to enhance a companies CMM level, introducing DSDM will help CMM level 1 organizations to achieve at least level 2, since a DSDM implementation will result in a repeatable project structure. The same is true for ISO 9001 in its most basic process maturity model, however ISO 9001 provides far more complex process quality then CMM, but organizations certified for ISO 9001-3 can rely on the DSDM manual or other sources to learn how DSDM and ISO 9001 can coexists. [BSI03]

7.2 Extreme Programming

While Extreme Programming has been the first well known methodology to handle agile software process, it is perfectly suitable to integrate into a DSDM implementation, since many concepts of DSDM can improve XP with a robust requirements and project management mechanism. [CALD02]

8. Conclusion

The DSDM framework is a straight forward framework based on best practices to start implementing a project structure, its strengths being simplicity, extendibility, proven in the past but not claiming to be the solution to all kind of projects. The fact that the very authors of DSDM issue a warning concerning the single-minded use of DSDM where it does not suite the purpose provides much more credibility to the concept then many other similar methodologies.
The weakness of DSDM is, like with many other structured approaches, the relatively high barrier to entry (apart from the licensing costs). Switching to DSDM is neither cheap nor fast, and requires a significant cultural shift in any organization, because suddenly deliverables are replaced with tasks. I suppose for many people accepting the fact that their project will be on time and on budget, but not necessarily delivering the initially requested functionality will be hard to accept.
DSDM clearly presents itself as the most mature agile development method, while many agile methodologies are rather programming methodologies then process models. The select few agile methodologies share common ground with DSDM, ie. SCRUM which as well as DSDM, promotes team empowerment. The Crystal Methodologies in turn assume to be a good collection of best practices as well, where as DSDM does a much more comprehensive job in showing its evolving character by versioning their framework after every revision by the DSDM consortium. [ADM04]

Shell, Loyds Bank Insurance Services, British Telecom, British Airways, Deutsche Bahn, Hewlett-Packard, Renault, the city of Los Angeles; the list of companies using DSDM is long. Many other organizations may already use DSDM, but are not yet ready to publicly commit to DSDM, even the worlds largest software vender, Microsoft, borrows ideas from DSDM when publishing his own semi-agile solution framework currently in version 3.1 complete with 25 000 certified practitioners.
Agile methodologies are very visibly on the rise, as well as another promissing concept, Offshoring. Xansa used DSDM to organize its offshore development in India; ThoughtWorks has been using offshoring to India with agile methods as well, proving that the highly profitable practice of offshore development and onshore engineering works with agile methods, good enough to present a business case. Further research into agile methodologies and offshoring could provide actionable evidence for business which IT and development efforts they should relocate and whether agile methods are supporting the change. [FRAZ03], [CALD04], [SIMO02], [THEE03]

[SUTH03] Sutherland. Web 2003 http://jeffsutherland.com/scrum/

[COCK02] Cockburn, Highsmith, Jones. Web 2002 http://alistair.cockburn.us/crystal/crystal.html

[DSDM04] DSDM Consortium Frequently Asked Questions. Web 2004, http://www.dsdm.com/en/resources/faqs.asp

[DSDM04a] DSDM Consortium Official Presentation. Web 2004 http://www.dsdm.com/kss/download.asp?fileid=444

[YEBO83] The Relationship among Group Size, Yetton, P.W, Bottger, P.C, 1983. http://citeseer.nj.nec.com/context/1701780/0

[BSI03] British Standard Institute, DSDM & TickIT. Web, 2003 http://www.tickit.org/dsdm.htm

[CALD02] Caldow, DSDM Consortium, DSDM and XP. Web 2002 http://www.dsdm.org/kss/download.asp?fileid=66

[ADM03] Advanced Development Methods, Web 2003 http://www.controlchaos.com/Case3.htm

[FRAZ03] Frazackerly, DSDM Business Focused Development p. 131 ff. 2003

[CALD03] Caldow, Distance No Object: Working Offshore. Web 2003 http://www.dsdm.org/kss/download.asp?fileid=325

[SIMO02] Simons, InformIT. Web 2002 http://www.informit.com/content/index.asp?product_id=%7BA6DD6BC5%2DB290%2D4CC5%2D9F24%2D67E8A683E36D%7D

[THEE03] The new geography of the IT industry, The Economist. Print/Web 2003 http://www.economist.com/displaystory.cfm?story_id=S%27%29HH%2EQA%5B%21%23%40%20D%0A