

Large Projects

Table of Contents

1	INTRODUCTION.....	2
1.1	AIM.....	2
1.2	AUDIENCE.....	2
1.3	CONTRIBUTORS.....	2
1.4	DEFINITIONS AND TERMINOLOGY.....	2
1.4.1	<i>Project or Programme?</i>	2
1.4.2	<i>Defining Size</i>	3
2	CAPABILITY AND EXPERIENCE.....	5
3	APPLYING THE DSDM PRINCIPLES.....	6
4	THEMES	16
4.1	STANDARDS, PROCEDURES AND AGREEMENTS.....	16
4.2	ORGANISATION AND CO-ORDINATION	16
4.3	COMMUNICATION.....	17
4.4	ARCHITECTURAL FOUNDATION.....	17
4.5	MOMENTUM.....	17
5	BREAKING DOWN LARGE PROJECTS	18
5.1	PARTITIONING.....	18
5.2	SEQUENTIAL SUB-PROJECTS	19
5.3	CONCURRENT SUB-PROJECTS	19
5.4	RISK REDUCTION	19
6	CONCLUSIONS	20
7	GLOSSARY.....	21

1 Introduction

DSDM is being used on projects that are very large indeed. This White Paper investigates the changes in management (project, risk, quality, configuration, etc.) that using DSDM on such projects brings.

During production of this White Paper large projects were debated widely. Inevitably a proportion of this debate related to large projects in general, independent of method or approach. To ensure that this paper reflects the objective of the Task Group, the principal focus has been on *the application of DSDM to large projects*. However, it is not possible to completely separate this topic and there are elements of the paper that apply to large projects in general.

1.1 Aim

The objective of the White Paper is to provide advice and guidance on the effective application of DSDM to large projects.

1.2 Audience

The audience for the White Paper is both business and IT project management who are considering embarking on a sizeable DSDM project.

1.3 Contributors

This White Paper was put together by a Task Group reporting to the Technical Work Group. The paper has been produced as a result of a series of workshops and from written contributions submitted by the Task Group members below.

David Cooper	Logica
David Lightwood	Sema Group (Task Group Chair)
Chris Maxwell	CMD Corporation
Chris Read	IBM

The paper was updated for conformance with DSDM V.4.2 in 2003 by Jennifer Stapleton, Independent Consultant. Note: Some of the original paper is now part of the standard content of DSDM.

1.4 Definitions and Terminology

1.4.1 Project or Programme?

The first question is “What is a large project?” Before attempting answer the question, the distinction between large projects and programmes must be clarified.

A **programme** will be characterised by the following:

- it is concerned with implementation of a business vision;
- it includes a diversity of activities;

- projects will be created and destroyed;
- it may be open-ended with respect to time.

A **large project** on the other hand will be characterised by the following:

- it contributes to implementation of the business vision;
- it has a specific focus/scope - associated with some aspect of the business vision;
- it has a well-defined timescale and scope.

In addition a large project can often be further characterised by the following attributes:

- comprises packages, bespoke development, systems integration, legacy;
- utilises mixed development methods;
- incorporates multiple deliveries;
- includes non-software components - e.g. cultural change.

The focus for the Large Projects Task Group is on large projects rather than programmes. However, this should not preclude applying the principles to programmes in the future.

1.4.2 Defining Size

There are many different ways of defining the size of a project; some of the more commonly used ways are:

- function points
- effort
- budget
- number of entities, processes, etc.
- number of objects
- duration.

For the purposes of the Large Projects Task Group we have defined large as being 'outside the DSDM envelope'. To understand what this means in practice (and, in particular, to quantify it) we have analysed the attributes of what might generally be regarded as a suitable project for DSDM on the basis of its size. Thus we have a number of indicators which point to a project being outside the DSDM envelope:

- more than 5 developers/testers per team;
- more than 5 development teams;
- more than 1,000 function points per development team;
- more than 6 months elapsed time.

These measures are not absolute. In practice there will be a continuum from the smaller projects which fall well within the DSDM envelope through projects which are on the

boundary of the envelope to projects which are significantly outside the envelope. The number of thresholds which are exceeded and the extent to which they are exceeded will give an indication as to where a project is on this continuum - i.e. how far outside the *DSDM envelope* the project is and to what extent the issues which are addressed in this paper will be significant to the project.

2 Capability and Experience

In embarking on any systems development project an organisation must understand its capability and relevant experience in relation to the needs of the project. DSDM projects are no different in this respect and the question which should be asked is not simply “How large is it?”, but “How large is it in relation to our capability to deliver?”

In considering this issue an organisation must ask itself a number of questions, the answers to which should all be ‘yes’, namely:

- has the organisation got a track record of successful project delivery, irrespective of method?
- has the organisation successfully completed projects of a similar size?
- has the organisation successfully tackled projects comprising multiple and different sub-projects?
- has the organisation successfully used DSDM before (on smaller development projects)?
- has the organisation got relevant capability with respect to the technical environment in which the system will be developed?
- does the project satisfy the criteria for suitable applications for DSDM? In particular, does the project satisfy the Suitability Filter criterion specifically targeted at larger projects, i.e. can the application be split into smaller functional components?

The purpose of asking these questions is to minimise risk to the project by establishing that the only significant aspect of the project which is new or represents an area of uncertainty for the organisation relative to other projects is the ‘scaling up’ of DSDM.

N.B. Reference to the ‘organisation’ in the above implies both the business and the development entities who will jointly undertake the development; they may both be from the organisation or be drawn from separate organisations, e.g. where a third party is employed to provide IT skills.

The above are in addition to the Critical Success Factors contained in the Suitability Filter which apply irrespective of the size of the project.

3 Applying the DSDM Principles

What follows is a discussion of the issues that might be faced in applying the DSDM principles to large projects.

Active user involvement in rapid application development is imperative

Number of users

The role of the users on the project is much the same as for a smaller project. However, the larger number of user representatives involved demands that more attention is paid to co-ordinating their involvement especially in the context of the larger user community which they represent.

A large user community often means greater diversity of users and increases the difficulty of finding key user representatives, in particular Advisor Users of whom more are likely to be involved.

Continuity of user involvement

The longer elapsed time of many large projects (and the need for users to be involved over an extended period) may result in a tension between the needs of the business and the demands of the project with senior management being reluctant to commit to the level and duration of involvement demanded. The project is at risk if staff are swapped in and out of the project (either core development team staff or key user representatives). This is due to the time lost for the newcomers get up to speed, the relative lack of ownership and different outlook that will accompany the arrival of any newcomers.

All people involved in the project must be fully aware of the duration of their involvement and be able to commit to it as a means of maintaining continuity.

On a very large project comprising a number of related sub-projects the creation of a core team of business and technical staff with involvement across several sub-projects and multiple phases of work can provide a basis for continuity of knowledge as the project progresses.

Senior management commitment

A large project will almost inevitably demand the involvement of more users and for an extended period of time. It is essential that senior user management are aware of this extended user involvement and are committed to sustaining it for the duration of the project. Even where the project duration is not extended there will be a need for a more substantial level of involvement that will require the commitment of senior management.

The senior user representatives on a large project (Executive Sponsor, Visionary and Ambassador Users) are likely to be key people in the user organisation with many calls upon their time making it correspondingly more difficult to sustain their involvement in the project at the desired level.

Knowledge of DSDM

The increased number of users (and developers/testers) involved in a large project increases the chance of some of the team not fully understanding the DSDM approach. Particular effort must be made to ensure that everyone involved in the project understands the method sufficiently well to be able to undertake their role(s) effectively.

On large projects comprising both DSDM and waterfall developments, there is the possibility of causing confusion to individuals who move between teams over the life of the project (see the Managing Mixed DSDM and Waterfall Projects White Paper for some guidance on this).

Location of the team

The bigger the project the less your chance of collocating with your end-user partners, especially if they are geographically dispersed. The most extreme solution to this issue is to rent office space and move everyone in. Another approach is to locate the development team in an end-user area, or at least make sure that the development team visits the users (rather than making the users visit the development team).

N.B. It may not be necessary for all sub-projects to be colocated. However, the members of each individual sub-project team must be colocated, if at all possible.

There are a number of ways in which the effects of a dispersed team, using enabling technologies including workgroup products and e-mail, as well as more traditional technologies such as telephone answering machines, voice-mail, etc.

DSDM teams must be empowered to make decisions

Dissipation of empowerment

DSDM relies on empowerment to drive the decision-making process. The number of people involved in a large project will mean it is harder to ensure empowerment. Executive or Board level sponsorship will be required to commit to the empowerment of representatives of large numbers of users.

In small, well-focused developments, users can reasonably be expected to act as representatives of all, and make decisions for all. However, on a large, diverse project with a wide user community, each user will feel representative of only a small part and will be more reluctant to make decisions that affect others. If not managed properly, this will potentially increase the amount of time spent on decision-making. Defining the scope and terms of reference for empowerment for each user representative is one way of ensuring effective empowerment in this respect.

To avoid unnecessary debate and referral to authorities outside the project only those people who are adding value should be involved in the process (on a larger project there will be more marginal stakeholders which could have an adverse effect on empowerment and the decision making process). Ensure that the Ambassador Users on the project command respect amongst their peers so that their decisions and input carry weight.

Escalation

These issues cannot be addressed by just saying that users “must be empowered”. In large projects, it may be unreasonable to draw together *all* the representative users, and so clear escalation lines and procedures are essential so that when users feel unable to represent everyone, they can escalate the question to higher authority quickly. Allowing users to escape from responsibility when necessary will allow them to exercise their empowerment more freely when they feel able.

Arbitration

A fallback strategy is to adopt a policy of arbitration for decisions that cannot be reached within an agreed time frame. Appoint an arbitrator, ideally a member of the user community to expedite the decision-making process and, if all else fails, to make the decision themselves. It is imperative that the arbitrator is decisive. It is one thing to empower someone to make decisions: it is a completely different thing to make them decisive. It is important that the project recognises that the arbitrator's decision is final.

Other steps that can be taken to reduce this effect are:

- adopt a no blame culture so that people are not afraid for their jobs or position if they make a bad decision
- an agreed business (and, to a lesser extent, technical) architecture will provide a context for the resolution of issues
- ensure cross-team visibility of issues, decisions and their impact
- foster a one-team mentality re-inforced through a clear understanding of common goals.

Scope of empowerment

The increase in the number of people working on the project and, in particular, the existence of multiple teams demands that there are clear guidelines defining the scope of empowerment. These must be explicit about the extent of the changes that a team can make without consultation with authorities outside the team, e.g. the Project Manager or DBA.

The focus is on frequent delivery of products

Start-up

On a large project, the need for a higher degree of formalisation and infrastructure represents a start-up delay that will tend to militate against early delivery. This must be managed by adjusting the scope of the deliveries, especially the early deliveries.

Managing the deliveries

Match the rate of delivery to what the business can accept and manage the expectations of the business through prioritisation in relation to the business issues, risks and benefits. A Prioritised Requirements List in which the functions to be delivered are prioritised

according to their value to the business will assist in planning deliveries, especially in maintaining a business focus to the deliveries.

A release policy and plan is needed on a large project to formalise the process of delivery into the business. This policy must address the issues described above, providing a common context for implementation for the potentially larger number of parties involved in or affected by the implementation.

If the project has phased delivery, the team (both development and users) will be needed to support the implementation phase of the lifecycle, as they are the only ones that understand the system, potentially diverting key resources which are needed for the next phase of development.

Sustaining momentum

Whilst it is reasonable to expect the team to work intensively for the duration of a small project, you cannot expect this sustained level of commitment for projects with a duration much longer than three months.

The role of the Project Manager comes to the fore in this context. If the project is to succeed the project manager must have excellent management and soft skills in order to get either the best from a bigger team or from a smaller team over a longer development timescale.

Fitness for business purpose is the essential criterion for acceptance of deliverables

Business architecture

A business architecture is essential to provide a common business context for the project in terms of the business vision, strategy and KPIs. The business architecture clarifies the scope of the project in business terms by identifying both the business processes that are to be supported and the information required by those processes. It also defines the business benefits they will deliver.

Inter-dependencies

The complex inter-dependencies between deliverables produced by different teams over a period of time and often concurrently will constrain the absolute ability to meet the business needs. Hence the need for a business architecture that reflects this business need and that forms the common point of reference for all development teams on the project. Thus, on a large project, it is likely that the business architecture will form the basis for acceptance.

Iterative and incremental development is necessary to converge on an accurate business solution

Scope creep

The increased size of the project team and, in particular, the number of development teams will increase the risk of scope creep. Consequently, effective formal change

management procedures are essential, particularly for changes that extend beyond the boundary of a single sub-project.

The rigorous application of timebox principles will also help focus the teams on their objectives and reduce the tendency to allow requirements to grow.

Technical environment

Supporting iterative and incremental development on a large project presents a number of issues relating to the technical environment:

- the increase in the scale and complexity of the project demands that tool support is adopted more comprehensively
- the tools used must be capable of scaling to support the size of development team planned
- the tools must be able to produce scaleable application components
- the number of different tools is likely to be greater on a large project. Wherever possible tools which integrate well should be selected
- standards will be required covering the use of the various tools employed in order that they are used consistently across the development teams
- a strategy for the technical environment is required to ensure compatibility of tools and to address the need to deploy applications across differing platforms and technical environments
- guidelines on how to use the tools in an iterative and incremental development environment will be required
- the increased number of developers and testers on a large project coupled with a possible increase in the number of tools being used to support various aspects of the method increases the likelihood that some of technical staff will not be familiar with all the tools at the start of the project. Adequate provision must therefore be made for training and overcoming the learning curve as well as seeding those developers and/or testers who are familiar with the tools throughout the team.

Inter-dependencies

The increased number of inter-related and inter-dependent components on a large project will potentially put constraints on both:

- the degree of freedom which the development teams have to allow the iterative process to converge on the business solution
- the release of increments of the system to the business.

To address this it is essential to have both a business and a technical architecture that provides:

- the framework within which iterative development can take place, clearly defining the bounds within which there is freedom to develop a function
- the basis for an incremental release plan that reflects these inter-dependencies (which in turn will be reflected in the Outline Plan, the Development Plan(s) and the Implementation Plan(s)).

Related to this is the increased importance of the role of Technical Co-ordinator on a large project. The scope of this role covers quality (standards and procedures, etc.), configuration management and senior technical authority (system architect or chief designer). The scope of this role on a large project is such that it will probably be distributed amongst several people, each with a specific area of responsibility, but who work together to provide a single co-ordination function to the project.

All changes during development are reversible

Configuration Management

There will be more configuration items that need control in a dynamically changing environment. A formalised approach to configuration management is essential to both understand and manage the impact of reversing changes and to identify the inter-related components that must also change in support of any one change.

The scale of the problem demands that tool support for configuration management is adopted. Ideally this should take the form of a repository that is integrated with the other tools in the development environment.

The corollary of this is that there is increased scope for reuse on large projects. For this to be realised there must be a policy supported by appropriate procedures that reflect the need to pro-actively manage reuse, if it is to deliver the benefits it promises.

Inter-dependencies

The potentially complex inter-dependencies between components will almost certainly impose some constraint on the extent to which changes can be reversed. Even if the problem can be managed using configuration management tools, there will be practical limits to which the project can accommodate the impact of changes (in terms of the timescale and cost) when they involve many components with complex inter-dependencies.

To compensate for this reduced flexibility to reverse changes it is important that the need to reverse changes is reduced to a minimum. This will be achieved by having formal business and technical architectures that form a clearly defined framework within which development can proceed in a way that is less prone to the need to backtrack.

Decision-making strategy

The increased complexity and impact of any changes will mean that more time is needed for the decision-making process. To keep this to a minimum the approach to change must

be formalised. There must be a well-defined strategy which provides a framework and guidelines to facilitate rapid decision-making which:

- does not unduly constrain the inherent flexibility of DSDM
- protects the overall integrity of the system and its implementation timescales and plans.

Requirements are baselined at a high level

Uncertainty

Due to a number of factors, principally the sheer scale of the project and relative familiarity of the larger number of developers, testers and users with the method, there will often be a temptation to baseline requirements at a lower level to reduce uncertainty. This fundamentally goes against the principles of DSDM and the size of the project must never be used to justify a move to the more formal and detailed documentation of requirements that is seen outside DSDM.

Architecture and interfaces

A clearly defined technical architecture which reflects how the high-level requirements are to be implemented is essential to provide the framework or context for iterative and incremental system development on a large DSDM project (in much the same way as it is on a smaller project). The size of the project does not demand that the architecture is defined to a lower level of detail. The only exception to this is in the definition of the interfaces between sub-systems where a lower level of detail is necessary in order to provide sufficient understanding that changes to the interface definition can be avoided.

DSDM or waterfall?

The broader scope of a large project increases the likelihood of there being some parts of the system that are not suitable for a DSDM approach. Project management must therefore anticipate and plan from the outset for parallel developments using both DSDM and traditional waterfall approaches.

To avoid the need for changing methods mid-way through development, the requirements must be understood sufficiently well to be able to determine the right approach for each sub-system. This is fundamentally no different from DSDM as applied to smaller projects where the choice of method is made in principle during the Feasibility Study and validated during the Business Study and Functional Model Iteration.

Testing is integrated throughout the lifecycle

Integration testing

The importance of integration testing increases on a large project with many sub-systems and multiple and frequent deliveries. The size and structure of the team is such that there will probably be a need for a separate integration testing team.

A strategy for testing is required which reflects the importance of integration testing and defines how developed modules are to be integrated and tested within the overall system (parts of which may already be live) such that they cannot have a de-stabilising effect on previously released modules. The scope of integration test will potentially include non-software modules, e.g. hardware, training, new organisation structures. It is important to start integration testing early and to maintain consistency throughout the project. The people involved in test planning should include project infrastructure and support staff (who will be responsible for the operational system) as well as those performing the DSDM Tester role.

Architectures

The importance of architectures is highlighted in integration testing. They provide the basis for validating that assembled modules and sub-systems support the overall business processes and requirements:

- functional testing of assembled modules against the business architecture
- non-functional testing of assembled modules against the technical architecture.

N.B. Testing of the individual modules, both functional and non-functional, will have been completed prior to integration testing as part of the development of those modules.

The nature of testing will reflect the objective of ‘fitness for business purpose’ insofar as it will be concerned with verification or validation that the business requirements have been met (as opposed to producing an audit trail demonstrating that ‘everything works’).

Testing Tools

The increase in the amount of testing which will be required on a large project (principally in integration testing) increases the scope for using automated testing tools and the value which may be derived, especially when applied to regression testing.

Acceptance testing

Acceptance can fundamentally follow the principles of DSDM as applied to smaller projects with progressive acceptance distributed throughout the lifecycle:

- functional acceptance of modules during the Functional Model Iteration
- non-functional acceptance during the Design and Build Iteration
- acceptance of a sub-system (or release) during integration testing as part of the Implementation phase.

A collaborative and co-operative approach between all stakeholders is essential

Universal buy-in

The increased number of stakeholders involved in a large project increases the probability of there not being universal buy-in to the approach. This increase in the number of stakeholders is not simply a function of the greater number of developers, testers and user

representatives on the project itself, but the larger size of the project will raise its profile in the organisation, attracting the attention of a wider range of stakeholders, some of whom will almost certainly be senior people in the organisation.

Arbitration

On larger projects there is more scope for differences of opinion and interpretation to emerge, especially where an external service provider is developing the system. An arbitration procedure must be in place to quickly resolve any such differences before they have a negative impact on the project. The contract between service provider and customer is definitely not the vehicle for arbitration. Indeed, if ever either party resorts to quoting parts of contracts, the Project Manager will have failed and the project will almost certainly be doomed to failure.

One team mentality

The increase in size of the team on a large project will make it increasingly difficult to instil a 'one-team' mentality to the project. There will inevitably be personal agendas that emerge from time to time. This extends beyond the project team itself and includes stakeholders external to the core development team who may be looking to the project to serve their own ends, which may not be consistent with the objectives set for the project.

The Project Manager must be aware of this and be both pro-active in preventing it from manifesting itself on the project and quick to identify and deal with any instances of it.

Open approach

Adopting an open approach to project management and keeping all stakeholders fully informed about the issues the project faces and the impact of decisions taken will help foster an atmosphere of trust between all stakeholders. This will ensure that there are no surprises - one of the biggest causes of a breakdown in relations between the various parties involved in projects.

Contractual

A large project implies a large amount of money and therefore (in perception at least) a high risk for both developer and purchaser. Thus the contractual difficulties found in introducing DSDM are likely to be exaggerated, with a correspondingly lower likelihood that either party will accept a vague contractual position. This extends to the pressure to use standard, well-understood development processes to reduce the risk.

A further characteristic of large projects is that they are likely to involve more departments, including those specifically responsible for commercial and contractual negotiation. This makes building commercial trust between supplier and purchaser more difficult, impacting directly on one of the key DSDM CSFs highlighting the importance of 'a supportive commercial relationship'.

There are any number of contractual alternatives, for example: shared risk/shared return, function point delivery or fixed price. However, the most important consideration is to

ensure that whatever the basis for the contract, the business commits to evolutionary development and the contractual relationship reflects this.

4 Themes

Throughout this paper are a number of recurring themes. These are described below. Closer examination of these themes reveals a common thread: *formalisation*. A relatively informal approach may work successfully on a small DSDM project (it could even be described as a characteristic of the approach). However, it will not necessarily be appropriate on a large project where the increase in scale, principally in terms of people, will demand a higher degree of formalisation. The key is to apply just sufficient formalisation to overcome the issues of scale without compromising the principles and benefits of DSDM through over-bureaucracy.

4.1 Standards, Procedures and Agreements

The relatively small number of people involved in a small DSDM project means that sharing knowledge of how things should be done can be accomplished relatively easily, i.e. informally. However, the number of people on a large project demands that standards, procedures and agreements are formally documented as follows:

- *procedures* covering project management, change control, configuration management, estimating, escalation, training, etc.
- *standards* describing the approach to modelling, development, system test, HCI style, etc.
- *agreements/SLAs* covering user involvement and availability, response times (to questions or issues raised by the project team) and escalation.

4.2 Organisation and Co-ordination

Again, principally due to the number of people involved, a large project demands increased organisation and co-ordination within teams, between teams and between the project and the business:

- *organisation structure* The organisation structure must be formalised and include clearly defined roles and responsibilities in order that it is well understood what each person on the team is doing. A large project may require a number of temporary or transient organisation structures over its lifetime.
- *co-ordination* There will be a larger number of activities and deliverables with complex inter-dependencies which will require careful co-ordination; this will demand a rigorous approach to progress reporting.
- *continuity* both of developers, testers and users over an extended timescale, both within a single project and across multiple related projects is a key success factor.
- *DSDM competency centre* can act as the 'broker of best practice' across the project
- *project office* to act as a central point of reference for the project and facilitate consistency in information sharing.

4.3 Communication

Further reflecting the number of people involved in a large project (both within the project team and from the wider business community), there is a need for a very high level of communication to provide visibility of issues, decisions and their impact to everyone involved or affected by the project, namely:

- within and between teams, taking account of the increased geographic dispersal of users, developers and testers where this is a factor
- between the business and the project
- between related projects
- with the wider user community.

The language of communication should reflect the various types of audience (business, technical, development) and highlights the importance of soft skills.

4.4 Architectural Foundation

An architectural foundation is important on any project, but it becomes essential on a large project in order to provide a common context for the project. There are a number of facets to the architecture, the two principal ones being:

- *business*, both high-level based on the business vision, business objectives and KPIs to provide the focus, direction and rationale for the project and the lower-level view defining the business environment, processes and information relevant to the project (on a large project of extended duration this lower-level business architecture may change reflecting changes in the business)
- *technical* On a large project there will be several teams working on different parts of the system at any one time. The technical architectures (data, function, communication, interfaces and development environment) provide a common framework within which individual teams can develop separate parts of the system in a way which still enables them to integrate them into a coherent whole. The number of discrete developments being undertaken on a large project demands that special attention is paid to interfaces.

4.5 Momentum

Large projects demand a greater focus on maintaining the momentum of the project over an extended period of time to keep both the development team and the business interested and motivated. DSDM is inherently well positioned to address this through the principle of regular and frequent deliveries based on clear, short-term goals and by recognising and celebrating the achievements and success of the project.

The approach to project management should re-inforce this by adopting a results-oriented approach (or even management by objectives) rather than a task-oriented one.

5 Breaking down large projects

One of the most practical means of addressing a large project is to break it up into smaller, more manageable sub-projects. The following provides some guidance on how this might be achieved:

- prioritise functions within the application on the basis of the value they are expected to deliver to the business to produce a 'value stack'
- aim for a low level of functional binding to enable parallelism in development
- use the value stack to drive the Development Plan and to construct work packages, each of which is self-contained, has a well-defined scope and builds on the previous work packages. Each work package should be contained within a timebox of no more than 6 months duration
- construct a release strategy based on a grouping of work which reflects the implicit value of the work packages as derived from the value stack and the ability of the business to use the functionality
- assign the work packages to development teams. Adjust the size and profile of individual teams within overall DSDM guidelines to reflect the needs of the work package (especially the timebox). The work packages will provide a clear framework for project management.

5.1 Partitioning

A technique which may be useful in defining the work packages is partitioning. The objective must be to select a basis for partitioning which minimises the interaction between the work packages - making the interfaces simpler and the issues of co-ordination, communication and organisation as straightforward as possible. Approaches to partitioning include:

- *business process* (for example a planning project, a distribution project, a customer service project)
- *process/data affinity* Analysing the process and data models of the business architecture to identify clusters of processes which use the same group of data objects
- *technical platform* Where more than one technical platform is to be deployed, partition the project around the different platforms
- *architectural* Use the architectural components of the system as the basis for partitioning
- *organisational* Where there are several distinct user groups associated with the system, it may be beneficial to partition the development around these groups.

The Development Plan that is derived will potentially result in the need for sub-projects to be run sequentially (and delivered in a phased implementation) or concurrently (and delivered in a single implementation) or a combination of both. It should also highlight non-IS sub-projects.

5.2 Sequential sub-projects

If time permits, this is the preferred way of breaking a large project down into smaller more manageable sub-projects. Breaking a project down into smaller sub-projects which are scheduled to run sequentially offers the following key advantages:

- the overall team size can be kept down to the optimum size for a DSDM project;
- as each sub-project is implemented there is an opportunity to review the remaining requirements to see if they are still needed;
- if the same team is retained for subsequent sub-projects their experience, both of the business and the project, will increase with an attendant increase in productivity.

There is, however, a price to be paid. The obvious fact that the early phases of the project will deliver products which need to be maintained means that there is an extra change control and configuration management overhead.

5.3 Concurrent sub-projects

If time does not permit the project to be broken down into smaller, sequentially phased sub-projects, an alternative approach is to run several sub-projects in parallel. Whilst this offers the advantage of compressing the development timescales there are some disadvantages:

- a larger overall team size (maybe composed of several sub-teams) is required which introduces problems of communication and control
- more co-ordination is required between the teams both in the technical aspects of the project and the business aspects
- the parallel developments may adopt different approaches (i.e. DSDM and waterfall).

5.4 Risk reduction

A further approach to breaking down large projects, but more specifically focused on reducing the inherent risk associated with large projects is to adopt a staged approach as follows:

- *proof of concept* to verify that the project will deliver the desired result
- *pilot* to prove the technical environment and approach - a pilot team starts ahead of the others in order to identify and solve problems and then pass this experience on to the teams following behind (the pilot team will be made up of the more experienced practitioners complemented by consultants from the tool vendor)
- *phased implementation* based around sub-projects identified along the lines as described above.

6 Conclusions

The most significant conclusion to emerge is that there is no fundamental reason why DSDM cannot be used on large projects. However, the decision to use DSDM or not will always be a subjective one and will differ according to the experience of the people undertaking the project. It is therefore impossible to put exact criteria for excluding DSDM from a project.

The DSDM Consortium has already acknowledged the validity of applying parts of DSDM to projects: *you can use all of DSDM most of the time and some of DSDM all of the time*. It follows that even if it is decided that it is not appropriate to apply DSDM in its entirety on a large project, there will almost certainly be opportunities to apply elements of the framework successfully, e.g. a subset of the principles, timeboxing, prototyping, JAD, etc.

In summary, if you are about to embark on a large project using DSDM, address its size by focusing attention on the key themes of *formalisation, organisation and co-ordination, communication, architectural foundation* and *momentum* whilst managing the application of the nine principles.

7 **Glossary**

CSF

Critical Success Factor

DBA

Data Base Administrator

HCI

Human Computer Interface

IS

Information System

KPI

Key Performance Indicator

QA

Quality Assurance

SLA

Service Level Agreement