DSDM® CONSORTIUM

DSDM and UML

Table of Contents

1 Introduction

Software development projects rely heavily on diagrammatic models to bridge semantic gaps between business and IT team members. Table 1 describes concisely how DSDM (a software development approach without prescribed modelling techniques) and the Unified Modelling Language (a modelling notation and model-driven technique without a prescribed process) complement each other.

DSDM	UML
DSDM is a process framework without a prescribed notation.	UML is a notation without a process.
DSDM is non-prescriptive: it says what you need to do to run a successful RAD project, but it doesn't tell you how to do it. It permits you to choose the most suitable techniques for a particular project and to adapt the framework to different circumstances using "all of DSDM some of the time and some of DSDM all of the time".	UML can be adapted to suit the circumstances: it is non-prescriptive; you can just use whichever techniques are most suited to the particular development, extending the notation as required; UML acknowledges that different processes will be suitable for different projects.
DSDM was designed to embody best practice in RAD.	UML was designed to bring together the best features of different OO modelling techniques.
DSDM emphasises the importance of delivering both users' functional requirements and their non-functional requirements, including ease of use and maintainability.	UML, as an OO modelling language, supports the solicitation of users' needs in models that are easy to use and easy to maintain. UML is not well suited to capturing non-functional requirements.
DSDM demands productive developers and enhances developer productivity.	UML may enhance developer productivity.
DSDM requires an iterative and incremental approach.	UML has been designed with iterative and incremental development in mind.

Table 1: DSDM and UML synergy

1.1 Aim

The White Paper has the following objective

- To illustrate the sort of products that you can expect to see in a DSDM UML project
- To show which UML products enhance the DSDM products and phases.

1.2 Scope

The guidance in this White Paper applies before and during a project of the following nature:

- a business process project (i.e. a project with no technical development content)
- building an Enterprise Architecture
- a software development project (and not necessarily in an object-oriented development environment).

1.3 Audience

This White Paper supplements the content of the DSDM Manual version 4.1. It does not attempt to teach UML. Basic familiarity with both DSDM and UML is assumed. Readers who wish to get some understanding of UML and its origins should refer to the reading material in the bibliography. The White Paper is based on UML version 1.4. The White Paper is aimed primarily at the following people:

- Project Managers
- Developers
- Business Users
- Business Architects
- Technical Co-ordinators

1.4 Section Structure

Following the introduction in section 1, section 2 provides guidance on introducing and evolving the UML products in the DSDM phases.

Section 3 discusses the DSDM roles and responsibilities in an UML environment.

Section 4 contains some brief concluding remarks.

The Appendix contains a bibliography.

1.6 Related White Papers

The DSDM consortium has produced a number of White Papers that may also be useful when considering a DSDM and UML project:

- The DSDM Development Techniques White Paper would be of assistance when making a choice about which modelling technique to use. It includes guidance on object-oriented techniques.
- The DSDM and Component Based Development White Paper is a guide to producing the components required for a CBD project and re-using them.
- The Guidelines for Reuse in DSDM White Paper highlights where reuse improve the speed and accuracy of delivering systems in a DSDM environment.
- The Process Prototyping in a DSDM Project White Paper describes techniques that can be used to get Use Case detail (see section 2.2).

2 DSDM Phases and UML Products

2.1 The table

Table 2 details the UML components that could be introduced or evolved within each of the five DSDM phases from the project lifecycle. To supplement this, the case study on the website contains examples of the level of detail that can be expected in each stage of the lifecycle.

Table 2: UML in DSDM phases

UML Notation	Feasibility Study	Business Study	Functional Model Iteration	Design & Build Iteration	Implementation
Use Cases	Major Use Case names	Complete list of Use Cases. May have a definition of major Use Cases. The Prioritised Requirements List is based on the Use Cases.	Normal' usage of the Use Case (ie what happens 80% of the time). Drives the prototyping process.	Descriptions of Use Cases, refined as needed to ensure fitness for business purpose. Now includes the exceptions as well as the main flow. Drives testing.	Used to build User Documentation. Support departments should receive these UML products
Interaction (Sequence/Collaboration) Diagrams		Could be used when creating a Business Process Model/Enterprise Model to help understanding	One sequence diagram per major Use Case. Component diagrams for individual scenarios. More if required. Forms part of the Functional Model	Part of Design Prototype. Structured sentences derived from sequence diagram description will drive testing.	Support departments should receive these UML products
Class Diagrams	Major problem domain concepts can be captured as an initial set of classes. These could be extracted from a logical data model or the class diagram from an earlier project.	A first cut business class diagram forms a useful element of the Business Area Definition; should be process-driven, derived from the Use Case definitions and Interaction Diagrams	Business Class Diagram is product of the Sequence Diagram and Use cases (all part of the Functional Model)	Business Objects are fairly stable at this stage. Add application/system control here (could draw a parallel with logical/physical database design)	Support departments should receive these UML products

State Transition Diagrams			Has additional notation which may be useful to describe business processing. Additional processing of relevant/selected Business Objects. Performed at class level and only where it can add value.	As in Functional Model Iteration but now concentrating on control and communication objects	Support departments should receive these UML products
Component Diagrams		A project in a mature OO environment could be given a set of components at this stage and therefore use this diagram to show scope. Used in the System Architecture Definition, a few important components will be shown here.	A collection/partitioning of the Business Classes. The 'What'. Should contain all the proposed components		Support departments should receive these UML products
Deployment Diagrams	Used for scoping in an environment with a stable software/hardware architecture	Relevant to the System Architecture Definition.			Part of the Delivered System. Support departments should receive these UML products.

Activity Diagrams	Used anywhere to support the other UML products. Especially useful for Business Process Modelling	Used anywhere to support the other UML products. Especially useful for Business Process Modelling	Used anywhere to support the other UML products. May be useful to describe Business Processing. Additional processing of relevant/selected Business Objects. Performed at class level and only where it can add value.	Used anywhere to support the other UML products. Used to make test scripts	Used anywhere to support the other UML products. Support departments should receive these UML products.

2.2 Producing the UML Products using DSDM techniques

At the beginning of a Project when starting with a blank sheet of paper (=Pre Project and Feasibility Study)

Within the Feasibility Study, perhaps as part of the project launch workshop, participants could take part in a facilitated workshop to identify all of the principal actors, Use Cases and classes that will be needed. A sensible mix of users and developers is required. The facilitator should have the attributes listed in the role description of the DSDM Manual. The scribe would ideally be experienced in UML modelling, and therefore able to sketch UML models on flipcharts and ensure everyone involved understands them. Flipcharts and/or electronic whiteboards are useful in these sessions, but it is perhaps too early to introduce a modelling tool into these workshops. A follow-up facilitated workshop could use CRC cards (Class, Responsibilities and Collaboration – see *Designing Object-Oriented Software,* details in the Bibliography) and role-playing to test what has been produced, refining and correcting as necessary. Defining and refining Use Cases is done best in an iterative way. Both the number of Use Cases as the number of Classes can give an first indication on the required amount of effort needed to develop the system.

Even during Pre Project UML notations like Use Cases and Activity Diagrams can be used for an outline scope for the investigation to take place during the Feasibility Study.

To get detail on the Use Cases (= Business Study, Functional Modelling Iteration and Design and Build Iteration)

A further workshop might be valuable for detailed Use Case definition. Each participant must be a stakeholder in the system. Storyboarding may be an active way of drilling down to what actually happens/should happen within each Use Case. In addition, the *Process Prototyping in DSDM Project* White Paper contains some techniques that may also be useful when fleshing out Use Cases. One of the key advantages of Use Cases from a DSDM point of view is that they can be used to derive partitioned work-packages that can be prioritised, developed and tested to build the final solution. These work-packages can be used to define increments. The dependencies between Use Cases must be taken into account when defining the increments. A Use Case can be related to other Use Cases by Extends and Uses relations.

Use Cases can be tested by verification and validation. Verification of working code after code generation. Validation early during the project by step wise walk through of the Use Case. This can be done by role-playing; participants take on the role of an actor or a specific system part.

Per Use Case several Test cases (both logical and physical) should be developed during Functional Modelling Iteration to test the Functional Prototypes and the Design Prototypes during Design and Build Iteration. Developing activity diagrams can be an alternative for the textual description of a Use Case. The actors will help to identify the affecting user groups (part of the Business Area Definition in Functional Modelling Iteration).

The detailed Use Case description can be developed by the Ambassador Users, this ensures that the description is in the language of the users and recognisable. Object Constraint Language (OCL) of UML can be used to document additional information that can't be recorded in the Class Diagram. These documents and models complement the Functional Prototype and are part of the Functional Model.

Component Diagrams and Deployment Diagrams and their use in the System Architecture Definition (= Business Study)

These diagrams represent the physical aspects of the system under development. As such they are both natural candidates for inclusion in the System Architecture Definition (SAD), if this level of detail is required. Both are forms of class diagram, with the classes substituted by icons representing specific elements of the system.

The component diagram shows the internal structure of the physical software components such as executables, libraries, tables, files and documents. A component diagram is typically used to:

- model source code
- model executable releases

- model physical databases
- model adaptable systems.

The deployment diagram models the hardware structure of the system and the runtime allocation of software executables to the processing nodes. A deployment diagram is typically used to:

- model embedded systems
- model client/server systems
- model distributed systems.

Prototyping and the System Architecture Definition in Functional Model Iteration

When the project progresses into Functional Model Iteration, a prototype should be developed to test the integrity of the SAD and reduce the technical risk. Quality criteria for the prototype would be based around "do all components communicate with one another as required?"

The architecture is defined by several views, each necessary to complete the overall picture. These views are:

- Use Case view (Process)
- Logical view (Class diagram)
- Concurrency view
- Component view (Software Components)
- Deployment view (Hardware / Platform)

Documentation in Design and Build Iteration

During Design and Build Iteration the development of User Documentation, Training Documentation and other documentation accompanying the Tested System is started. Use Cases and Activity Diagrams can help to structure and develop this documentation.

3 DSDM Roles in a UML environment

3.1 Roles and Responsibilities

Project team members and project roles have a many-to-many relationship. Roles and responsibilities are defined in the Development Plan. All roles in the DSDM manual are also valid for projects using UML. Where they are responsible for developing specific products, this must be refined by the corresponding UML models. The development team is responsible for developing and maintaining the UML models (see Table 2). The following section covers the additional Specialist Roles.

3.2 Specialist roles

Specialist roles exist to fulfil a specific function or bring specific detailed knowledge to the project. The following are likely to be needed to support the use of UML (they are defined in the *Component-Based Development* White Paper):

- **Project Librarian:** A Developer or a Scribe may undertake this role. He or she is responsible for managing the UML library, in particular the storage of UML deliverables in the shared repository, version control of UML deliverables and controlling access to the repository.
- **Component assessor:** with knowledge of available component libraries and their potential use in the project – refer also to the *Guidelines for Reuse in DSDM* White Paper
- **Component librarian:** adds approved components to the component library and manages their use
- **UML expert:** specialist modeller with knowledge of UML modelling techniques.

3.3 Responsibility Matrix

The matrix in Table 3 shows responsibilities and products and which role is responsible for them.

Table 3: responsibility matrix

	Developer	Ambassador User	Advisor User	Technical Co-ordinator	Component Librarian	Component Assessor	UML Expert	Project Librarian
Responsibility determine UML deliverables for each DSDM stage **Product:** UML work plan, in System Architecture Definition				✔			✔	
Responsibility: provide knowledge of specific business processes		✔	✔					
Responsibility: contribute to review and acceptance of UML models **Product:** validated models		✔	✔		✔	✔		
Responsibility: develop required diagrams as defined for project: **Product:** validated models	✔	✔					✔	
Responsibility: use diagrams in building code **Product:** code	✔							
Responsibility: maintain UML diagrams through lifecycle **Product:** validated models under version control	✔							
Responsibility: manage UML project library **Product:** UML project library								✔
Responsibility: define standards for use of UML **Product**: UML standards and ensure use				✔			✔	

Responsibility: oversee use of UML and ensure use of component libraries in project **Product**: Ensure use of UML standards and component libraries				✔				
Responsibility: provide UML consultancy and guidance **Product:** UML consultancy				✔		✔	✔	
Responsibility: manage component library **Product:** managed component library					✔			

4 Concluding Remarks

Despite their different origins, DSDM and UML are strongly compatible. In order to use UML techniques within the DSDM framework, it is necessary to utilise the UML products in the appropriate phases of DSDM.

For developers with an OO background wishing to exploit the benefits of DSDM this White Paper has provided guidance on adapting the DSDM lifecycle.

Supplementary guidance on techniques and clear worked examples are available on the members' pages of www.dsdm.org.

This paper has also identified additional roles and responsibilities related specifically to OO projects, which would be using UML.

DSDM relies on tools, techniques and people skills to ensure the semantic gap between user and developers is addressed. UML, when used and explained correctly, can improve DSDM projects by assisting them to close this gap effectively.

Appendix A – Bibliography

"Joint Application Design": August, J., Yourdon Press, 1991, ISBN 0-13-508235-8

OMG Unified Modelling Language Specification (http://cgi.omg.org/cgi-bin/doc/formal/00-03-01)

This is the official UML Version 1.3 specification. It is downloadable from the above site (pdf format, approx. 5 MB). Includes Summary, Semantics, Notation Guide, Extensions and Object Constraint Language. The Notation Guide is the most practical for day-to-day use; the Semantics section mainly of interest to academics and Case Tool manufacturers.

"The Unified Modelling Language User Guide": Booch, Rumbaugh and Jacobson: Addison Wesley: ISBN 0-201-57168-4

This book is the definitive guide and tutorial to UML and its use by the original authors of the language.

"The Unified Modelling Language Reference Manual": Rumbaugh, Jacobson and Booch: Addison Wesley: ISBN 0-13-087014-5

This book provides detailed definitions and examples of all terms in the UML notation.

"UML Distilled": Fowler: Addison Wesley: ISBN 0-201-32563-2

Arguably the best book to have by your side during early UML projects. This book presents a practical, concise overview of the UML; do not be fooled by its size (168 pages).

"Use Case Driven Object Modelling with UML": Rosenberg: Addison Wesley: ISBN 0-201-43289-7

This book describes a DSDM like development process using UML. Although a slim volume, Rosenberg takes a pragmatic approach to the need for documentation and describes pitfalls to avoid at each stage in handy checklists.

"Analysis Patterns: Reusable Object Models": Fowler: Addison Wesley: ISBN 0-201-89542-0

This is an ideal book for Business Analysts and those setting standards to be used within DSDM projects. Fowler describes many reusable OO/UML patterns found across business domains that should decrease the time taken to produce the Business Area Definition.

"Design Patterns: Elements of Reusable Object-Oriented Software": Gamma, Helm, Johnson and Vlissides: Addison Wesley: ISBN 0-201-63442-2

Further to the Analysis Patterns (above), this book describes 23 patterns that are representative of good OO design which would reduce risk and increase the productivity of DSDM developers; also available on CD in HTML format.

"Designing Object-Oriented Software": Wirfs-Brock, Wilkerson and Wiener: Prentice Hall: ISBN 0-13-629825-7

Information about the use of CRC.

"Writing effective Use Cases": A. Cockburn: Addison-Wesley ISBN 0 201 70225 8

Writing Effective Use Cases describe how "actors" interact with computer systems and are essential to software-modelling requirements. For anyone who designs software, this title offers

some real insight into writing use cases that are clear and correct and lead to better and less costly software.

Contributors

Steve Ash, Independent – OO Training & Consultancy; Rob Day, Independent - RDA Limited; David Harrison, Popkin Software; George Hay, CMG Admiral; Vic Page, London Metropolitan University; Jim Rook, Centre for Software Engineering Limited; Jennifer Stapleton, Independent; Gwen Young Chair), Independent.

Thanks are also due to Clare Casson of COI Communications, Kevin Day of HPD Software, and Amanda Kent of DHL for their specific contributions. Thanks for updating the White Paper to Rik Jan van Hulst of The Vision Web, Kees Peeters (Independent) and Raimond Wets of Cap Gemini Ernst & Young.

