Joint Defra/EA Flood and Coastal Erosion Risk Management R&D Programme

Software Requirements for Joint FCERM R&D Programme Modelling

Outputs and Architecture Specification for RASP Family Outputs

R&D Technical Report FD2121/TR1

Environment Agency

defra
Department for Environment Food and Rural Affairs

Joint Defra/EA Flood and Coastal Erosion Risk Management R&D Programme

Software Requirements for Joint FCERM R&D Programme Modelling Outputs and Architecture Specification for RASP Family Outputs

R&D Technical Report FD2121/TR1

Produced: September 2007

Author: Dr. Jon Wicks, Halcrow

Statement of use
The report is intended to guide those involved in software development of joint Defra/EA Flood and Coastal Erosion Risk Management R&D modelling outputs.

Dissemination status:
Internal: Released internally
External: Released to public domain

Keywords: software requirements, RASP family architecture

Research contractor: Halcrow Group Ltd, wicksjm@halcrow.com

Defra project officer: Prof. Edward Evans

Science Theme Manager: Dr. Suresh Surendran

Publishing organisation
Department for Environment, Food and Rural Affairs
Flood Management Division,
Ergon House,
Horseferry Road
London SW1P 2AL

Tel: 020 7238 3000 Fax: 020 7238 6187

www.defra.gov.uk/environ/fcd

Published by the Department for Environment, Food and Rural Affairs (Sept 2007). Printed on material that contains a minimum of 100% recycled fibre for uncoated paper and 75% recycled fibre for coated paper.

PB No. 12794A

Acknowledgements: This document has benefited from the involvement of the following organisations: Environment Agency CIS, HR Wallingford, Wallingford Software, EdenVale Modelling Services, JBA, Wallingford HydroSolutions and Newcastle University.

Executive Summary

Software products form important outputs from many Flood and Coastal Erosion Risk Management (FCERM) R&D projects and will be used to help implement the Environment Agency's Flood Risk Management Modelling Strategy. For these software products to be readily useable by the Environment Agency, other operating authorities and their consultants, it is important that the software adheres to relevant software standards. The FD2121 project has developed guidance material to assist research contractors in understanding and conforming to the relevant standards with an emphasis on Environment Agency standards. In addition, the project has reviewed the software modularity of the RASP family of decision support tools and has initiated documentation of common modules and certain enabling architecture.

The primary source of relevant standards for FCERM software is the Environment Agency's Corporate Information Services (CIS) 'Enterprise Architecture: Technical Reference Model' (TRM). Guidance material for R&D contractors has been developed from the TRM and from discussions with CIS staff and other industry experts. The guidance is presented in R&D Technical Report FD2121/TR2 'R&D Software Development Projects – Guidance for Research Contractors'. In addition to the objective of guiding the R&D contractor towards producing conforming software, the guidance documentation is also designed to foster early informed discussions between the R&D contractor and CIS.

The utility of the guidance has been demonstrated through three trial applications: GLIM-CLIM rainfall generator, MDSF2 and the NFFS model adapter. These represent examples of classes of software ranging from background university R&D (rainfall generator), through projects focussing on delivery of new methods to operating authority and consultant staff (MDSF2), to specific commercial modelling software development for Agency systems (NFFS model adapter). The development of the guidance has highlighted a number of areas which could be addressed by improvements to the TRM or other process documentation, these include: the need for more guidance on .NET, the need to facilitate end user involvement in the development stage, improved documentation requirements, and the need for early consideration of future custodianship, support, maintenance and user training.

The RASP family of decision support tools has been described in the report 'Scoping the development and implementation of flood and coastal RASP models' (SCO50065/SR1, 2007). The scoping report identified a range of RASP-based bespoke decision-specific tools which, although targeted at different FCERM business functions, share common data and modules. The requirements and methods for the RASP family of tools continue to be developed and currently are not sufficiently well defined to enable a comprehensive and appropriate conceptual/logical architecture for the RASP family to be fully identified. However useful steps towards an appropriate architecture have been made in this report covering: an architectural review of ongoing RASP-related projects (NaFRA, MDSF2, RACE, PAMS and CRUE),

currently identifiable common modules and appropriate enabling technologies. The review has shown that the tools are being designed to share common data and some common computational modules. Further action is required in the areas of defining requirements, analysing commonalities and further specification of software architecture to better achieve the objectives of facilitating the efficient production of sustainable and appropriate software tools and to facilitate competition. As requirements and methods continue to evolve it will be important to review architectural aspects and maintain an on-going dialogue with CIS and other Agency/Defra managers to facilitate take up of the software outputs.

Contents

1 Introduction

1.1 Project Objectives

Defra's Flood Management Division funds a joint R&D programme with the Environment Agency to support Flood and Coastal Erosion Risk Management (FCERM) within England and Wales. The flood and coastal management policy aims to reduce risks to people, property and the natural environment from flooding and erosion.

Amongst the important outputs from the joint R&D programme will be a considerable number of software products. Some of these will be operational or planning tools which will be mounted on Environment Agency systems. They will therefore need to conform with Environment Agency standards on architecture, languages, software platforms (e.g. GIS systems), interface and data exchange protocols, testing and acceptance.

A second point is that a number of them will be inter-related. The RASP (Risk Assessment for System Planning) family is perhaps the prime example, but there are parallel systems being scoped or developed in other fields such as the NFFS (National Flood Forecasting System) and tools for the WFD (Water Framework Directive). For efficient development and support these should be designed so as to use common modules wherever possible, within an open software architecture.

A third point which follows from the first two relates to competition and access to the best ideas from a range of originators. This will be greatly assisted by the availability of a document which sets out clearly an open, common set of protocols and architecture for the RASP family.

The objectives of this research project were thus to address these points as follows:

Objective 1: To draw up and agree with Environment Agency IT specialists a common set of requirements covering standards on enterprise architecture (languages, software platforms, interface and data exchange protocols), testing and acceptance, potentially applicable to all software outputs of the joint R&D programme, in order to permit fair tendering and the efficient production of conforming software.

Objective 2: To draw up for the RASP family an overall system architecture which will identify and specify common modules. This will be declared openly in order to facilitate competition and the support and updating of the various applications within the family.

At the time of writing, the Environment Agency is drafting a Flood Risk Management Modelling Strategy. The Strategy aims to ensure that the Agency is able to achieve its current and future modelling obligations to support flood risk management in an effective and efficient manner. Key messages from the Strategy of direct relevance to the objectives of the FD2121 report are:

- The need for a risk based approach
- The need to be consistent with the Agency's IT Strategy and preferred enterprise architecture
- The need for modularity to facilitate reuse of components

The FD2121 objectives are consistent with the Flood Risk Management Modelling Strategy and the uptake of the findings will support implementation of the Strategy.

1.2 Purpose and Outline of the Report

This report provides the formal technical report detailing the outputs of the research project. The background to the development of the guidance and initial user feedback is provided in Chapter 2. Example applications of the guidance are provided in Appendix A. Chapter 3 covers Objective 2 (RASP family architecture).

In order to facilitate ease of use and future updating, the actual guidance for research contractors covering Objective 1 is provided in a separate report: R&D Technical Report FD2121/TR2 'R&D Software Development Projects – Guidance for Research Contractors'.

2 R&D Software Requirements

2.1 Introduction

One key deliverable from many FCERM R&D projects is software products. Historically these software deliverables have not been readily usable by all members of the anticipated user communities. A particular problem has been access to the software deliverables by Agency technical staff. A further issue has been that the software may not have been developed using 'best practice' approaches which may result in difficulties in taking the software forward after completion of the R&D project. The R&D project reported herein was conceived to help address these issues through the development of guidance on a common set of software requirements for FCERM R&D projects. The Agency's Corporate Information Services (CIS) provided key inputs during the development of the guidance.

The guidance has been developed through the following tasks:

- Gain a good understanding of CIS technical requirements for software which is to be installed on Agency systems (through discussions with CIS and a review of CIS documentation) (see section 2.2)
- Production of an initial draft guidance document
- Review of the draft document by CIS and by selected members of the anticipated user community (see section 2.4)
- Trial application of the draft guidance to three example software applications (see section 2.5 and Appendix A)
- Revision of the draft guidance in response to feedback from the review and lessons learnt from the trial application of the guidance
- Further review of the updated guidance by CIS
- Finalisation of the guidance to meet the requirements of CIS

The finalised version of the guidance is provided in R&D Technical Report FD2121/TR2 'R&D Software Development Projects – Guidance for Research Contractors', an overview of which is provided in section 2.3.

2.2 Understanding CIS Requirements

2.2.1 Overview

The documentation provided by CIS has been studied in detail. Key documents reviewed included "Overarching IS Principles" and "Enterprise Architecture: Technical Reference Model" (TRM). The TRM contains guidance, policy and rules to allow (amongst other things) software to be developed for the Agency taking into account CIS requirements. "Overarching IS Principles" is a set of 15 principles which are generally applicable, mandatory and underpin and reinforce Agency policies.

From review of the "Overarching IS Principles" and "Enterprise Architecture: Technical Reference Model" documents, it becomes clear that some of the

important themes to come out of those documents are that FCERM R&D software should be easily modified and extended, reused and supported (with minimum training requirements) – within the Agency infrastructure. The Agency will of course wish to get maximum value for money from FCERM R&D projects. As part of this, they will want software to be written in such a way that new features can be added easily and bug fixes can be done in isolation from the overall system as much as possible. To achieve these aims, software should be developed in a modular fashion, using loosely coupled functions, probably via Object Oriented development. Interfaces should be separate from business logic, a typical example would be the “Model, View, Controller pattern” (see http://java.sun.com/developer/technicalArticles/J2EE/despat/).

Certain FCERM R&D projects will develop software that contains functionality that it self will be useful for use in other FCERM R&D projects, or Agency developed software. Wherever applicable, it should be as easy as possible for a program to be created by one research contractor to make use of functionality from another contractor’s software, regardless of development environment. Maximum interoperability can be achieved by following the CIS “Enterprise Architecture: Technical Reference Model” and various methods such as creating “wrappers” around software, separating code into libraries, open communication and implementing data exchange via SOAP and XML. External interfaces and available functionality should be clearly documented.

Probably the single most important aspect of creating easy to support applications for the Agency is to create server side, thin client (browser based) solutions using Agency standard software platforms and development tools. If the user can use a browser then they have a head start in being able to run the software.

2.2.2 Agency platforms

Software developed to be run on Agency systems obviously needs to run on the platforms the Agency already uses (or will have at the time of delivery) and can support. An overview of the most pertinent information for FCERM R&D development follows (note that the “Enterprise Architecture: Technical Reference Model” should be considered the most up to date source for this information). When embarking on a project the expected timescale of development must be considered. In the case of a very short development timescale the current “Enterprise Architecture: Technical Reference Model” could be expected to be considered up to date and relied on. If, however, the timescale was much longer a discussion would need to be held with CIS to consider the possible changes to the CIS requirements over the period of development.

Applications should run over TCP/IP. Suppliers of proposed systems are to assume there is no available WAN bandwidth and should indicate the increase required.

Citrix is the Agency standard thin client enabling software.

Current server platforms are HP-UX, Novell Linux or Windows 2003. It should be noted that HP-UX is being phased out to be replaced with Novell/Suse Linux. The Windows 2003 servers are “hardened” builds specially configured to minimise security risks. All software development to run “server side” should run on Linux or Windows 2003.

Client machines run Windows 2000 (EA Build) v5 SP3, with Novell Client for Windows 2000/XP. These have a specially configured Internet Explorer 6 and not all components are available (for instance Javascript is allowed, but ActiveX components are not).

2.2.3 Application types

The Agency recognises that various types of software will be developed under the FCERM R&D programme. The type of software project being developed will have an effect on the application of the CIS “Enterprise Architecture: Technical Reference Model” standards and how rigorously they need to be followed. These can be separated into the following categories, though there may be some overlap and some applications may change category over time.

1. Basic Research & Development projects where the focus is on development of new methods that are “far from market” and the project specification does not require that the application is put on Agency systems. This will often be a “proof of concept” piece of software, which is likely to require much further work before becoming production software. As such, there may be scope for the “Enterprise Architecture: Technical Reference Model” to be applied less stringently. Obviously there are still good practices to be followed, which if followed will ease future transition to a fully-developed state.

2. Software developed primarily for non-Agency users, where the specification states it is not required on standard Agency systems. This type of software needs to take account of the varying systems in place at the operating authorities (and their consultants) when referencing the “Enterprise Architecture: Technical Reference Model” standards. For instance it is not practical to expect all consultants to install, run and maintain the standard Agency corporate database.

3. Software developed both for external consultant use and for use on standard Agency systems. This type of software needs to balance the requirements of the Agency with the practicalities of external consultants and other operating authority users being able to install and use the software.

4. Software developed solely to run on standard Agency systems. As such, of all the types of software this has to justify any deviations from the “Enterprise Architecture: Technical Reference Model” the most strongly.

5. Public facing systems (i.e. systems exposed to external use by the general public), have their own special requirements and costs. They are beyond the scope of this project and are not covered further.

2.2.4 Development Tools & Languages

In order to achieve the Agency's overall aims, there are standards for development tools. The most up to date standards can be found in the current "Enterprise Architecture: Technical Reference Model". At the time of writing (May 2006) the (applicable) standards are as shown in Table 2.1 below.

Table 2.1 Key CIS development standards

Area	**Standard Product (*subject to change*)**
Integration Hub	BEA Weblogic Integration
Analysis and Design Tools	Rational Rose Analyst Suite, Rational Unified Process
Development Tools	Borland JBuilder, Visual Basic (enhancement only) v6.0 (SP4), Oracle Forms and Reports (enhancement only)
Web Page Design Tools	Macromedia Dreamweaver v4.0
Architecture	J2EE for complex systems JSP & servlets for simpler systems
Application Server	BEA Weblogic Server v8.1
Web Server	Apache v2.0
Database	Oracle 9i v2
Browser	MS Internet Explorer
GIS	ESRI ArcView v8.2, ArcSDE v8.3.1, ArcIMS v4.0.1
Reporting Tools	Business Objects v5.0 (under review), Crystal Reports v7.0 (under review)

The Agency standards are for software development to be undertaken in Java. For Enterprise scale applications this is component based, n-Tier, using J2EE. The standard application server that is used is BEA Weblogic Server v8.1.

For smaller scale applications JSP and servlets (including the use of Apache Tomcat) are the standard development platforms.

Where the exact standards in the "Enterprise Architecture: Technical Reference Model" cannot be met, there are varying degrees of compliance (which require mitigating factors & evidence to be presented). For instance, although not the Agency standard, IIS (Internet Information Server) server side ASP (Active Server Pages) solutions with browser clients are preferable to Win32 client side executable code. Generally, for all applications there is a very strong preference for thin client web browser based architecture with rich client Win32 architecture only used where web browsed architecture cannot deliver the business requirements. The software would ideally work in any browser (to ensure

maximum usage by contractors and for future proofing), but obviously must work in the Agency standard browser software.

Fortran: The Agency understands that the Fortran language is still in widespread use and has its place (Fortran is fast, highly suitable for the sort of computational tasks required in FCERM R&D projects, there is large user base in the industry and a lot of pre-existing code). However, where Fortran is used, the Agency has a strong preference for Fortran "wrapped in Java" using XML, the hierarchy of acceptability of Fortran solutions in general is shown below:

1. Fortran "wrapped in Java" using XML (strong preference)
2. Fortran with XML inputs/outputs and control.
3. Fortran DLL (documented functionality callable by other languages)
4. Fortran solution

Java: Java is the Agency's standard development language. This is for both server side JSP or J2EE solutions and rich client solutions.

C++: For deployment of specialist applications requiring very high performance characteristics (e.g. modelling applications) which can not be achieved using the Environment Agency's general tool sets then the use of C++ may be justified. But bespoke code in general should be developed in Java.

C# / .NET: The use of C# / .NET would require strong justification and an ASP server side solution would be preferable to a rich client solution. It is worth noting that the Agency's application server solution has an adapter product which lets .NET functionality be called from Java.

VB: The Agency has some legacy applications written in VB, but VB is only supported for development in a maintenance role. As with any language other than Java, use of VB would need strong justification.

Other languages: There are a myriad of languages available, including emerging languages such as Ruby. The Agency has no specific policy on each and every language; the general principle is that Java is the language of choice.

2.2.5 Security

The main security issues that may be involved when considering FCERM R&D projects are:

- limiting access to applications where there is high processor usage and this needs to be managed
- limiting use to authorised users

Developers should bear in mind there is a management overhead for CIS if the new software implements its own security system, with users, passwords etc. This should be agreed with CIS as they will need to assign personnel to support this.

Data sensitivity is an issue. It is assumed FCERM R&D projects will not store information pertinent to the Data Protection Act – in the case that they do, CIS will need to be informed and data storage issues will need to be discussed.

When developing (server based) software it is also a good idea to have a management front end accessible through a browser client to perform administration, as it will be problematic for research contractors to get direct access to the host server. It is also extremely unlikely that any kind of dial-up access to the installed system will be allowed.

Where the Agency have purchased software it is expected that the application will not be locked down on the desktop for any reason after installation (for example, some applications require you to connect to the internet to verify the license).

2.2.6 Hardware Considerations

As part of any FCERM project it is important to consider whether there will be any hardware requirements as a result of installing the software. For example the bandwidth required by an application could mean that new hardware is required.

2.2.7 Migrating Software to Compliance

In updating existing applications or adding new modules there may be the opportunity to migrate towards a more compliant state. This might include writing Java wrappers around existing code, further modularisation of code, support of open format data exchange (e.g. XML). These types of changes would be supported by CIS, but obviously there may be cost issues and so cost versus benefit needs to be taken into account.

2.2.8 Databases

The current standard Agency database is Oracle 9i v2. As such, in general, new database based solutions are expected to be developed to work with this database. The Agency attaches high importance to this strategy. However, where solutions are required to run on machines outside of the EA infrastructure (e.g. contractors) the Agency realises it may not be practical or desirable to force contractors to install the database in order for the solution to run. The contractors may not be able to install it due to their own IT policy and systems, lack of expertise and cost (though it is noted that at this time there is a free version of Oracle available – Oracle 10g Express Edition). Where such issues arise it may be advisable to develop database agnostic solutions and as a general rule it will be advisable for developers to resist from using database platform specific features, such as stored procedures etc.

The Agency is unlikely to allow corporate databases other than their standard corporate database onto their systems.

There may also be situations when corporate databases are overkill for the task in hand or unavailable due to financial or personnel resources to buy and manage them. In these cases there may be limited situations in which MS Access databases would be accepted, but only with native access from within the application, i.e. MS Access not installed on the system. In extremely limited cases MS Access could be used via Citrix.

Embedded databases would need to have the ability to output their data to a common format (e.g. XML) as well as their proprietary format.

2.2.9 Non Database Data

Some programs will need some ancillary data that may not be suitable for database storage or the programs themselves will not require the use of a database. CIS would expect the developer to use XML as the format for both data and settings storage wherever possible. It is understood that when working with legacy/3rd party software that there may be a need to read and write from/to proprietary binary formats and that some of these formats are industry or de facto standards. However, when designing new software there would have to be extremely strong justification for using proprietary binary formats – probably the only example that could apply would be performance issues, but with availability of serializable XML etc., even this is doubtful. Where binary formats are proposed the Agency would expect to receive documentation as to the format of these and also expect some ability to handle/produce XML input/output.

2.2.10 Testing and Acceptance

The Agency use the “V” model of testing and will generally require research contractors to do the same. The “V” model is shown in Figure 2.1 which illustrates the clear alignment of the test stages with the planning and development stages. Agreeing the testing and acceptance strategy will form part of the agreement when a new project is begun and should be done before any test cases are written.

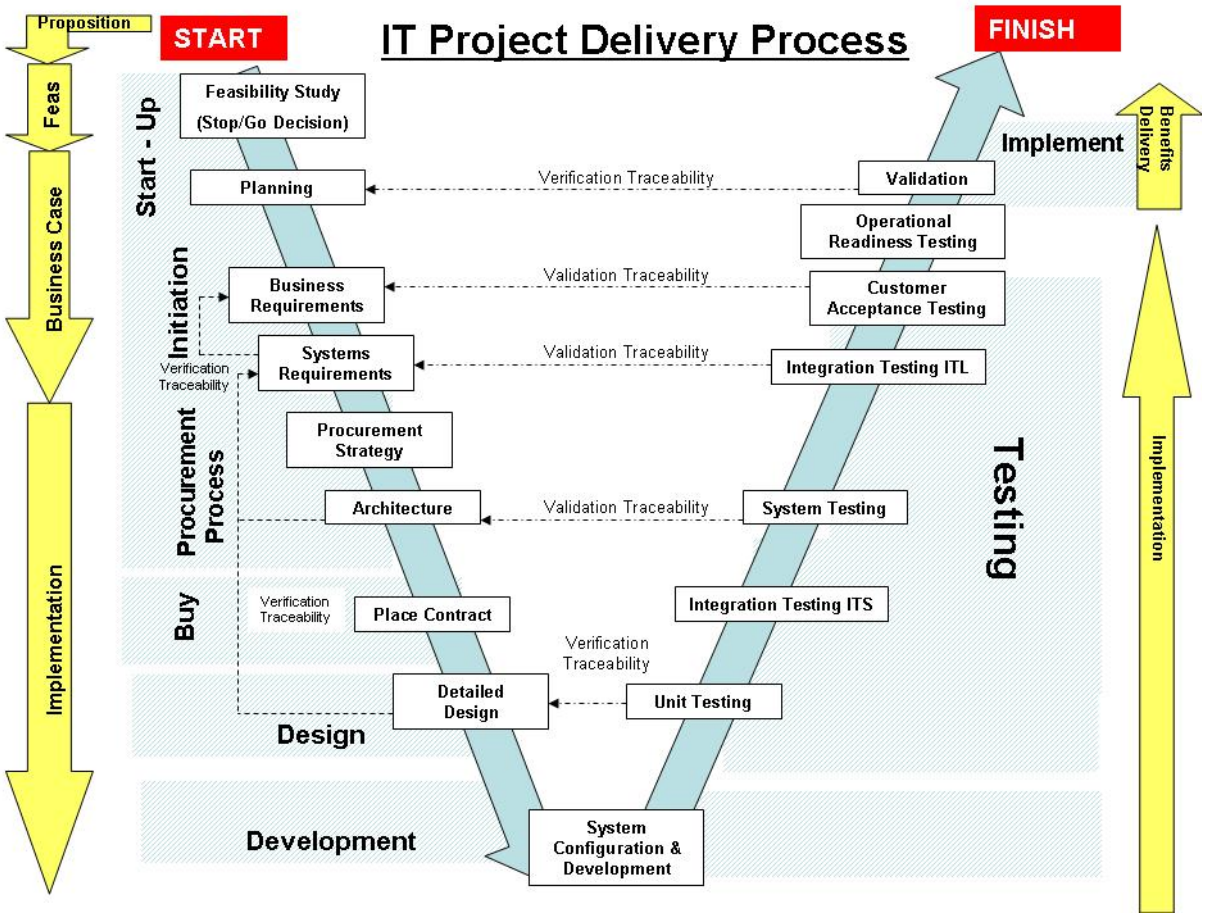


Figure 2.1 – IT Project Delivery Process including V-Model testing

When the Agency engages a 3rd party supplier, the 3rd party will take responsibility for the following testing stages:

Unit Testing: The objective of Unit Testing is to ensure that reliable program units are produced that meet their requirements and to identify errors in program logic. Typically, the developer who coded the unit will design and run a series of tests to verify that the unit meets its requirements. Each unit should be tested individually and in isolation by exercising its inputs and observing its outputs or behaviour. There are widely used tools (unit testing frameworks) available to assist in this task such as JUnit for Java and NUnit for .NET languages.

Integration Testing ("ITS-In The Small"): Components of code are assembled into sub-systems and linked to form a complete system. The objective is to test the relationship and links between individual units of code.

System Testing (inc. FAT - Factory Acceptance Testing): System Testing incorporates both functional and non functional testing. Functional system testing is focused on testing the system based on what it is supposed to do as defined in the functional requirements. Non-functional system testing looks at those aspects that are important yet not directly related to what functions the system performs. Non-functional requirements are just as important as

functional requirements and in all cases it is vital that these are tested prior to the launch of the system. It includes aspects like performance and security which are considered vital for today's web based applications.

System Test Process (Involvement by 3rd Party Suppliers): As the majority of projects will be developed away from an EA site by a 3rd party supplier this stage is likely to be the first opportunity for Agency Staff to see the application. To ensure quality, prior to delivery to the Agency, the 3rd Party supplier will be required to demonstrate compliance by hosting Acceptance Testing on their site, this will in effect be a pre-User Acceptance Test known by the Agency as Factory Acceptance Testing (FAT). This process will have defined, agreed acceptance criteria and will be subject to a Test Readiness Review Meeting prior to deployment on any Agency infrastructure. In addition, the 3rd party supplier will assist in the installation of the new application on the agency site. This will be done as part of the Acceptance Testing stage and will again require the 3rd party supplier to show compliance on the Agency's infrastructure.

Site Acceptance Testing (SAT): SAT will be used as a quality check to ensure that when the application is installed onto the EA infrastructure that it functions correctly with no critical errors. The 3rd Party supplier will be on site to assist and ensure that the application, specifically the server code, can be deployed correctly. The 3rd Party supplier will then conduct a subset of their System Tests to prove that the application can function without any critical errors. If possible, at this stage the opportunity should be taken for the 3rd party to forward any test assets such as test scripts that may be re-usable by the Agency.

When dealing with 3rd parties, the Test Management Process should also be carefully considered. The Agency Test Manager/Coordinator will decide if the project should produce a combined Test Strategy taking into account all testing stages or if separate strategies should be produced, one by the Agency and one by the 3rd party. This will also apply to Test plans and approach documents.

The 3rd party testing process should provide evidence of the tests run and results. In particular the Agency would like to see unit and system integration tests.

Testing methods other than the 'V' model would need justification and to form part of the project agreement. Software updates need to go through proper pre-production testing and the testing plan should be run against them.

2.2.11 Transition to Support and Maintenance

One of the key support and maintenance considerations is the decision as to who performs the support function – the supplier, CIS or both. The service level required for the support must be set as must any need for business lead users, database administrators etc. Further to this, developers will need to complete the most up to date "Service Support Requirements Brief" Agency document for each project that is to be handed over to the Agency. This is then used by CIS Support as the basis for planning operational requirements and estimating

support costs. It is preferred this takes place as early as possible in order that CIS can plan adequately ahead and spot any potential problems. Once the software goes into production, the document contents will generally be translated into a ‘Service Level Agreement’.

The main sections in the “Service Support Requirements Brief” are shown below:

- **Project Information** - General information about the project, key contacts, planned timescales and known risks and issues.
- **Overview of Application** – Description of the way the application is used within the business, its technology and functionality.
- **Support scope** – List of those components of the application which are in and out of scope for operational service and support.
- **Application Component Level Service Requirements** – Description of the requirements of the major IS components delivered.
- **Additional Support Services** - Description of any additional ‘special’ support services required. For Service Level Management and Supplier and Support & Maintenance use only.
- **Future Development Management** - Description of the services required to support future application developments and enhancements.

Specific supporting documents (such as Entity Relationship Diagrams) may need to be provided. These will have been previously agreed with the Agency Project Manager prior to formal product acceptance.

2.2.12 Software Deployment

Configuration and distribution of workstation files will be handled using a Novell NetWare Application Launcher (NAL) within Novell ZenWorks.

Suppliers of proposed systems must provide documented support for the application integration task to the standardised Agency desktop. This is performed using the ZENWorks for Desktops suite of tools which supports the use of Microsoft Software Installer (MSI) technology as well as its own built in application snapshot facility (SnAPPShot).

Where desktop software is to be installed on standard Agency desktops, CIS would expect the research contractor to detail the impact on the workstation, including the following:

- Assurance that the installation/application makes no changes to or deletions of protected operating system files.
- A list of dependent components (e.g. Active X controls, DLLs, drivers etc.)
- A list of Dynamic Link Library (DLL) and Application Programming Interface (API) calls.

- A list of known changes to registry keys

A lot of software is used within the Agency, on many machines, so the impact of new software on the desktop needs to be known and assessed. This is a major factor in the preference for browser based thin-client technology.

2.3 Overview of the Guidance

The structure of the guidance document (FD2121/TR2) is outlined below.

Chapter 1 – Introduction: This chapter explains the background, purpose and format of the guidance. It explains that the guidance document contains blank 'fields' which the research contractor should complete to document responses to the series of 'questions' contained in chapters 2 to 4 of the guidance. The chapter also contains a section highlighting the importance of entering into a dialogue with CIS (via a strategy analyst and/or project architect) – consideration and completion of the guidance document itself will not supply the required level of detail to confirm a satisfactory solution.

Chapter 2 – Pre Contract Award Guidance: This chapter is intended for use before the formal award of the main part of the research contract in order to gauge the level of CIS involvement required (and help identify any 'red flag' issues) early in the life of the project. Depending on the necessary level of CIS involvement identified, the Agency may wish to assign a strategy analyst and project architect to guide the contractor through the project. This will almost certainly be the case for any software to be installed on Agency systems.

Chapter 3 - Post Contract Award Guidance: This chapter forms the main part of the guidance and will steer contractors towards 'good practice' and facilitate early and informed discussions with CIS (where software is being developed for installation on Agency systems). The scope of the chapter is similar to the topics covered in section 2.2 of this report (understanding CIS requirements). Tables and decision trees are used to help present the topics.

Chapter 4 - Implementation Planning Guidance: This short chapter provides guidance for the preparation for implementation, for example, covering the need for support and maintenance (note that the activities of support, maintenance and training are not covered by the guidance).

2.4 Review of the Guidance

As stated in section 2.1, an early version of the guidance document was reviewed by CIS and by the following selected members of the anticipated user community:

- David Fortune (Wallingford Software)
- Chris Whitlow (EdenVale Modelling Services)
- Rob Lamb (JBA)
- Matt Fry (Wallingford HydroSolutions)

All reviewers were positive about the need for the guidance and agreed with the general approach taken in the draft document. The contribution to the development of the guidance made by the reviewers is gratefully acknowledged. The specific feedback from the reviewers is summarised in the table below (Table 2.2 for the research contractor comments and Table 2.3 for CIS comments). As noted in the 'response' column of the tables, nearly all of the suggestions made by the reviewers have been implemented in the version of the guidance contained in FD2121/TR2. In some instances it was decided to not implement the suggestions in the guidance and where appropriate these issues are discussed in Chapter 4 – Conclusions and Recommendations.

Table 2.2 Research contractor comments and responses

Reviewers Comment (on draft)	Response
Users will need access to the CIS document "Enterprise Architecture: Technical Reference Model".	CIS will need to make this available to research contractors.
Concerns that 'forcing' all software developed for use on Agency systems to be compliant may result in 'inappropriate' solutions or 'bypassing' of procedures.	Guidance has been altered to stress the importance of early and ongoing discussions with CIS so that the most appropriate solution architecture can be agreed.
Concerns that application of the guidance will increase the cost of some R&D projects.	There may be increases in the initial contract cost of some R&D projects but there are expected to significant benefits such as easier take-up of research outputs and lower 'whole life' costs.
Concerns that application of the guidance, and particularly necessary discussions with CIS, will impact on the programme for projects.	Guidance has been altered to recommend that time for discussions and agreement is built into the programme.
Concerns that using a 'thin client browser based' solution (the CIS preference) will present a step backwards in functionality and usability.	The guidance states the preference for a browser based architecture but acknowledges that an alternative architecture may be more appropriate if there is strong justification – dialog with CIS is recommended.
Recommendation that the guidance should acknowledge the gradual take-up of research outputs within software systems and the need to 'think ahead'.	Guidance has been updated to make this recommendation.
Recommendation that the guidance should encourage CIS to take non-Agency software users into account and to encourage Agency and non-Agency users to use the same software.	CIS already do take the whole project needs into account. Use of the guidance document will facilitate the understanding of the full project requirements and the relative importance of Agency and non-Agency deployment.
Recommendation that .NET applications should be more fully addressed (and supported) by the guidance and by the "Enterprise Architecture: Technical Reference Model".	This is an issue for CIS to consider further. The guidance has been updated to say that .NET applications may be acceptable with appropriate justification.
Recommendation that the guidance should encourage process staff and end-users to be	Whilst this is very good advice, it is considered outside of the scope of the guidance and

Reviewers Comment (on draft)	Response
involved in the development phase.	should be covered through other aspects of the R&D project.
There needs to be a mechanism to ensure that the guidance is readily available, promoted and used.	The guidance will be available through the Defra web site and R&D project managers should be encouraged to ensure it is applied.
Training, support, maintenance and upgrades are not adequately covered by the guidance.	Minor updates were made to the guidance to partially cover these topics – however full coverage was considered outside the scope as the guidance is intended to cover only the development phase.
Presentation and ease-of-use of the draft guidance should be improved.	The guidance was fully reformatted and areas for recording project information were made more distinct.
The guidance should state who to contact in CIS.	Names of CIS contacts have not been included in the guidance but job titles have been included.
Recommendation that the section on discouragement of proprietary data formats be changed to reflect the appropriate use of some de facto “standard” (but not fully open) file formats, e.g. GIS systems formats.	Guidance updated as suggested.
Recommendation to include mentioning the possibility of CITRIX-type access as an alternative to the preferred thin client browser based interface.	Guidance updated as suggested.
Removal of ‘Data formats in use’ topic and addition of items on ‘end user type’ and ‘security implications’ in the pre contract award guidance section.	Guidance updated as suggested.
Recommendation to include a ‘question’ on the use of coding standards.	Guidance updated as suggested.
Concern that predicting network bandwidth and processor usage is very difficult.	Agree – comment added to guidance acknowledging this but recommending that some information is provided as it is useful to CIS
Comment that it is too restrictive if software is not allowed to read and write to its own format files (provided they are not used by any other programs).	The guidance encourages the use of open (not proprietary) data formats such as XML even for data storage that no other programs are expected to use.
Concern over the potential negative implications of the requirement to produce all R&D software in Java.	Guidance updated to stress the need for strong justification if Java is not to be used.
Concern over the application security requirements in that they may enable unlicensed copying of software.	Guidance not altered. Specific concerns will need to be discussed with CIS during the project.
Concerns over the details of the software deployment process (installations being able to provide later versions of Windows system DLLs and concerns that developers will not have the	Guidance not altered. Specific detailed concerns will need to be discussed with CIS during the project.

Reviewers Comment (on draft)	Response
precise Windows build that the Agency use).	
Concern that XML will not necessarily provide the expected benefits when used for many data sets used in flood risk management. Issues identified include the need for well thought out schemas, inappropriateness of the tree structure of XML for multidimensional data sets, lack of random access to XML data and the verbosity of XML (leading to potentially very large files). HDF5 may be more appropriate for standard for many data sets.	Guidance not altered. Specific concerns will need to be discussed with CIS during the project.
Miscellaneous suggested minor changes to the text of the draft guidance.	Implemented where appropriate.

Table 2.3 CIS comments and responses

CIS Comment (on draft)	Response
The document needs to concentrate on ensuring the correct level of on-going engagement between the Contractor and CIS with the Guidance as a key part of the toolkit. The Enterprise Architecture is constantly evolving so projects will be at higher risk of non-compliance on delivery if the Guidance is taken in isolation.	Guidance revised to give more emphasis to ongoing engagement and the evolving nature of Enterprise Architecture
The report should conform to the documentation standard.	Document revised to conform to Defra R&D formats.
The report refers to specific products in the Enterprise Architecture (e.g. Technical Reference Model) – these should be treated as part of the overall guidance given by the Enterprise Architecture products. For example, the guiding principles of the IT Strategy and Enterprise Architecture are not mentioned but should be used from the outset to assess compliance. It should also be recognised that the Enterprise Architecture will constantly evolve and change – therefore, it is important to note in this report that contractors will need to acquire the latest version when embarking on a project.	Both points have been added to the "How to use this document" section
The report should highlight that the guidance provided does not constitute a full list of questions regarding the proposed solution architecture. If the project will deliver software that will be	This has now been clearly stated in "The purpose of this document" section This is covered in the new "Engaging with CIS" section.

CIS Comment (on draft)	Response
deployed within the Environment Agency, it will be necessary to enter into a dialogue with CIS, via the Project Architect, to fully review and understand what compliance to the Enterprise Architecture specifically means to each project.	
The engagement model (in particular with the CIS project manager and project architect) should be agreed and defined in this document. It is important that this aspect of engaging with CIS, as part of the overall engagement, is understood by the contractors. It is suggested that as projects pass the Idea and Proposition stages of the Improvement Cycle, the key contact is a Strategy Analyst (currently Stuart Pomeroy). A CIS project manager and project architect will be assigned to each project, where necessary.	This is covered in the new "Engaging with CIS" section although staff names are not used to facilitate 'future proofing'.
The document (section 3.2 onwards) allows assessment of specific aspects of architecture at a point in time – e.g. section 3.3 server processor utilisation communicated to CIS. Will this questionnaire be used: (a) as an instrument to kick off the dialogue between the contractor and CIS regarding Enterprise Architecture compliance? or (b) is it expected to maintain this document (e.g. reflect server processor utilisation communication to CIS completed?). If so, at what agreed point in lifecycle is it deemed "completed" and "signed off"? It is recommended that option a) will be adopted. The outcome of those discussions should be reflected in the normal project documentation (e.g. requirements specification or solution architecture) and be subject to normal sign-off.	Option (a) is recommended and this has been reflected with a paragraph near the beginning of section 3 to emphasise this.
Are the workflow diagrams intended to visually represent the relationships between the various individual questions? If so, it would be helpful if the questions were numbered and referred to in the workflow boxes.	There is no direct correlation and so numbering has not been added.
The report includes specific items from the Enterprise Architecture knowledge base (e.g. appendices A, B & C in draft version). Since these items, along with the rest of the Enterprise Architecture, is subject to continual review and change, it would be better to	Guidance changed as suggested so that references are quoted.

CIS Comment (on draft)	Response
reference external documents that will be maintained.	
Appendix D (in draft version) refers to some particular documents within our Enterprise Architecture – this is not a complete list. There are also other relevant guidance documents (e.g. overarching principles, information architecture principles etc.) which will be relevant for contract suppliers. It should be made clear that the relevant and up-to-date documentation will be available through engagement with the CIS (especially Project Architect).	Guidance updated as suggested (Appendix A and emphasised elsewhere).
How does the workflow diagram relate to the Environment Agency "Innovation Cycle"? To avoid confusion, it is important to map onto the overall process and ensure the steps and terminology map consistently.	Guidance updated to include "Innovation Cycle figure in the new "Engaging with CIS" section, which will give the research contractor an idea of where the various stages of the development fit in.
How does the workflow diagram relate to CIS "V" life-cycle model for system development? It is important to ensure consistency with terminology and map the steps/products within this process since stage checks are tied to this process. (In particular, test strategy and subsequent testing and user acceptance prior to system implementation will be relevant.) The V model is a useful tool even for small projects as a simple checklist. For larger projects it helps increase the probability of success.	Guidance updated to include the V-Model development cycle diagram.
Chapter 1: The contractor steps ("white" boxes) refer to processes which will need to be defined and agreed. Will this level of detail be produced as part of the FD2121 project? (Note: some of these process steps will need to incorporate and/or link with existing processes.)	It is intended that the FD2121 guidance will provide the contractor with an introduction to the tasks and processes that are likely to be involved and the flow of these. Any further definition of process will be undertaken as a project specific task and will be guided by the project architect.
Has the engagement model been discussed or described? This will determine how CIS (in particular a project architect) is appointed and his/her remit.	Guidance updated with new "Engaging with CIS" section. Note that the Agency R&D project manager/project executive will need to define the CIS involvement and, as the guidance is aimed at the contractor, we have not been prescriptive in this Agency-internal area.
Chapter 2: The table highlights some useful aspects of detailed development that would need to be reviewed from an Enterprise Architecture compliance perspective. It will	Attention has been drawn to this in the preamble (Chapter 1).

CIS Comment (on draft)	Response
seed the right sort of thinking and discussions. However, it should not be considered as "full and complete" since there will be project specific items which will need to be considered on a case-by-case basis.	
Chapter 2: There are no considerations given to security – is that intended? The aspects of security should cover system access by internal/external/standalone users (from intranet, Web etc.) as well as information security from data sensitivity perspective. Although these may be covered as specific considerations on the project – it would be useful to seed the notion of "thinking security".	Guidance updated as suggested to include an item in the Chapter 2 table on security.
Section 3.6: Suggest re-title the heading as Application Architectural Compliance	Guidance updated as suggested.
Section 3.7: As part of the list of justifiable reasons for exceptions, it mentions "skills not readily available in the organisation". What are the implications of this to the Environment Agency? Questions should encourage consideration for "total cost of ownership" and "essential business requirements" for the Environment Agency?	Guidance updated to remove "skills not available" from the list. Section 3.7 changed to emphasise "total cost of ownership" and "essential business requirements".
Section 3.9: There is mention of the Data Protection Act. However, there are other aspects of regulatory and legal compliance – the guidance is given by the Information Management Unit (IMU) via the Project/Enterprise Architect.	Guidance updated to include this comment.
Section 3.10: A paragraph on User Acceptance Testing should be included to complete the Testing programme.	Guidance updated as suggested.
Section 3.10: The functional specification should include a Test Strategy where any software is to be installed on Agency PCs. This should adopt the current principles, which are available on the Agency Easinet site and can be provided by the project architect	Guidance updated in Section 3.10 to include this requirement.

2.5 Trial Applications of the Guidance

In order to help develop the Guidance and demonstrate its use, it has been applied to three trial applications: the GLIM-CLIM rainfall generator, MDSF2 and the NFFS Triton Adapter. These represent examples of classes of software

ranging from background university R&D (rainfall generator), through projects focussing on delivery of new methods to Agency and consultant staff (MDSF2), to specific commercial modelling software development for Agency systems (NFFS Triton Adapter).

Appendix A sections A1 to A3 contain extracts from the R&D Technical Report FD2121/TR2 ‘R&D Software Development Projects – Guidance for Research Contractors’ partially completed for the three trial applications. The ‘highlighted’ cells in the tables contain the example information entered for the trial applications. Note that general figures and supplementary information contained in the guidance document have been removed to save space. To assist in cross referencing, the section numbering used in the appendix is consistent to that used in the guidance document.

It is important to note that these are not intended to be ‘approved’ examples which, if followed, would be acceptable to CIS – rather they show the type of information that is likely to be required to facilitate informed discussions with CIS.

3 RASP Family System Architecture

3.1 Introduction

The RASP (Risk Assessment for System Planning) family of decision support tools has been described in the report 'Scoping the development and implementation of flood and coastal RASP models' (SCO50065/SR1, 2007). The scoping report identified a range of RASP-based bespoke decision-specific tools which, although targeted at different FCERM business functions, share common data and modules. The requirements and methods for the RASP family of tools continue to be developed and currently are not sufficiently well defined to enable an appropriate conceptual/logical architecture for the RASP family to be fully identified. However useful steps towards an appropriate architecture can be made via an architectural review of ongoing RASP-related projects (NaFRA, MDSF2, RACE, PAMS and CRUE) to identify common modules and appropriate enabling technologies. Section 3.2 below provides an overview of the RASP-related projects and lists selected functional and non-functional requirements of the software tools expected to be delivered by these projects. Section 3.3 identifies commonalities and provides recommendations for aspects of their software architecture informed by software guidance described in Chapter 2 of this report.

3.2 Overview of RASP Family Functionality

The report 'Scoping the development and implementation of flood and coastal RASP models' (SCO50065/SR1, 2007) is a key document both in terms of explaining why RASP-based tools are essential for the Agency's business and in identifying future development requirements for RASP tools and methods. It is strongly recommended that the reader studies the scoping report before reading this chapter. However the scoping report does not (and was not intended to) provide detail on the required functionality (in software terms) of the RASP family of tools. The requirements are likely to evolve and become clearer as Environment Agency Policy and Process develop and as Science (R&D) develops and proves new techniques.

In order to start the process of identifying potential software commonalities within the RASP family a review has been undertaken of a sample of RASP related initiatives which are representative of:

- RASP-related tools in current use (NaFRA)
- RASP-related tools under development (MDSF2, RACE and PAMS)
- RASP-related initiatives about to commence (CRUE)

These RASP-related initiatives are introduced below and Table 3.1 identifies commonalities in functional and non-functional requirements. (Note that due to constraints on availability of information, budget and programme the classification of requirements contained in Table 3.1 should be considered as preliminary - it will require significant further work before it could be used for software design.)

NaFRA (National Flood Risk Assessment) - a self contained single "model" that implements RASP concepts to use or assess, at the national scale, selected source terms, defence performance, spreading of floodwater on floodplains and calculate selected risk metrics (including direct economic damage).

MDSF2 (Modelling and Decision Support Framework). MDSF2 is currently under development and will provide a desktop system for quantifying economic and social impacts of flooding for present and future scenarios with a range of flood management options. It is targeted at strategic planning (e.g. catchment flood management plans) but scale is not prescribed and therefore it can be used at a range of levels. MDSF2 builds on the first version of MDSF to incorporate new and improved risk-based methods implementing RASP concepts.

RACE (Risk Assessment of Coastal Erosion, together with Making Space for Water project HA4b - Risk Mapping Coastal Erosion) – application of the RACE probabilistic methodology to identify coastal erosion hazards for England and Wales. The project will include the generation of national hazard and risk data sets (through a bespoke software tool) and will facilitate local studies through a desktop tool. The relationship between these two broad requirements is similar to the roles NaFRA and MDSF2 take for flood impacts (i.e. national scale results and a separate tool for local studies).

PAMS (Performance-based Asset Management) – a programme of work to provide a system for assessing the whole life cycle of flood defence systems and provide an advanced method for decision making in terms of asset maintenance, renewal and new capital projects. The programme of work includes the development of RASP-based systems analysis tools with a user interface primarily designed for use by Environment Agency asset management staff.

CRUE component: "Effectiveness of non-structural flood risk measures" – a European collaborative research project which will develop and demonstrate an approach to simulating and assessing the long term effect of non-structural measures and their interactions, including land use planning, insurance, damage prevention, preparedness (e.g. early warning systems) and changed building practices (e.g. the use of stilts or bunds).

Table 3.1 Preliminary requirements list for selected RASP-related tools

Selected Requirements	Relevance for RASP-Related Tools:				
	NaFRA	MDSF2	RACE	PAMS	CRUE
Functional Requirements					
General file and project management	✓	✓	✓	✓	✓
Scenario management		✓			✓
Input and manage loading conditions	✓	✓		✓	✓
Flexible GIS-type processing capabilities		✓			
Fragility curve management	✓	✓		✓	
Point asset performance analysis		✓		✓	
Breach size representation	✓	✓		✓	?

Selected Requirements	Relevance for RASP-Related Tools:				
	NaFRA	MDSF2	RACE	PAMS	CRUE
Overtopping calculations	✓	✓		✓	?
System failure states analysis	✓	✓		✓	?
Flood spreading capability	✓	✓		?	✓
Coastal erosion hazard analysis			✓		
Coastal erosion risk calculation		✓	✓		
Represent structural responses	✓	✓		✓	✓
Represent non-structural responses		✓			✓
Receptor data management	✓	✓	✓?	?	✓
Economic impact calculations (annual damages)	✓	✓	✓?	✓	✓
Environmental impact calculations		planned			
Social impact calculations					
Risk to life calculations					
Calculation defence contribution to risk		✓			
Manage cost of responses				✓	
Benefit-cost calculation				✓	
MCA project appraisal calculations				✓	
Non-Functional Requirements					
Flexible user interface		✓			✓?
Run on Agency systems		✓	✓?	✓	
'Fast' run time (e.g. < 10 minutes)				✓	
3rd-party software independence important	✓?	✓	✓?	✓	✓?
Required to be 'open system'	✓?	✓	✓?	✓?	✓?

3.3 Recommendations for RASP Architecture

Where RASP-related applications are being developed for potential use by the Environment Agency then it is strongly recommended that the software architecture complies with the FD2121 guidance 'R&D software development projects – Guidance for research contractors'. Aspects of the RASP family architecture are described in this section under the following headings:

- logical component architecture (logical divisions of processing responsibility between blocks of code) – section 3.3.1
- physical deployment architecture (where components might be installed, e.g. database servers and client PCs) - section 3.3.2
- miscellaneous architectural/design aspects – section 3.3.3

As requirements and methods continue to evolve it will be important to review architectural aspects and maintain an on-going dialogue with CIS and other Agency/Defra managers to facilitate take up of the software outputs.

3.3.1 Logical Component Architecture

Table 3.1 provides a preliminary analysis of commonality of requirements within the selected RASP-related tools. A simplistic interpretation of Table 3.1 would be that if there is more than one 'tick mark' on a row then that particular

requirement is common and therefore should be implemented through a common module shared between the tools. Unfortunately this approach will not necessarily result in the most appropriate architecture. For example, while all tools will require general file and project management functions (e.g. opening and saving single files or collections of files) it is unlikely to be efficient to use a common module. It is more appropriate to analyse 'commonality' under the following categories:

- Common input data
- Common algorithms (equations and methods)
- Common software modules (e.g. shared dll's)

Table 3.2 provides a preliminary analysis of the functional requirements from Table 3.1 analysed in terms of these categories of commonality.

In the table a strong (mandatory) level of compliance with commonality is indicated by two ticks. Where there are probable benefits from commonality one tick is used. However, wherever there is commonality in algorithms then the presumption is that common software modules should be used – departures from this approach should only be allowed where there are strong reasons and with the agreement of the project board.

Table 3.2 Preliminary commonality assessment for selected RASP-related tools

Selected Requirements	Recommended level of commonality			
	Data	Algorithms	Software	Comments
General file and project management			✓ - see comment	Probably not appropriate to share code for this functionality (but there should be a common 'look and feel').
Scenario management	✓			A common 'language' (data model) for scenario components is recommended.
Input and manage loading conditions	✓✓			Use of common input formats (XML?) is recommended.
Flexible GIS-type processing capabilities			✓✓	Common software modules should be used where available.
Fragility curve management	✓✓	✓	✓	Common data formats and manipulation tools are recommended.
Point asset performance analysis	✓✓	✓✓	✓✓	MDSF2 and PAMS should share common data, algorithms and software modules.
Breach size representation	✓	✓	✓?	Currently implemented 'algorithms' are very simple and do not warrant shared software.
Overtopping calculations	✓✓	✓✓	✓✓	Common data formats and equations are recommended.

Selected Requirements	Recommended level of commonality			
	Data	Algorithms	Software	Comments
System failure states analysis	✓✓	✓✓	✓✓	More analysis required to confirm there is a benefit in sharing a software module.
Flood spreading capability	✓✓	✓✓	✓✓	Shared internal rapid flood spreading module(s) strongly recommended. To be implemented such that alternatives and/or enhancements can also easily be implemented. 'External' (e.g. commercial flood modelling software) software must also be able to be used.
Coastal erosion hazard analysis	?	?		Currently only relevant to RACE/HA4b.
Coastal erosion risk calculation	✓✓	✓✓	✓✓	Use RACE/HA4b erosion contours combined with common impact calculators. Likely to reuse some of the economic impact module.
Represent structural responses	✓✓	✓✓	✓?	More detailed analysis needed to understand how the tools will represent structural responses. Many structural responses may be modelled outside the RASP tool.
Represent non-structural responses	✓✓	✓✓	✓✓	Methods still need to be developed – they should be developed to enable use of shared date, algorithms and software modules.
Receptor data management	✓✓	✓✓	✓✓	Receptor data (e.g. property data) must be shared. Algorithms for 'adjusting' data (e.g. 'correcting' floor areas) should be shared.
Economic impact calculations (annual damages)	✓✓	✓✓	✓✓	A shared module for direct property damages is strongly recommended. As algorithms for other economic impacts are developed and proven then they should also be shared.
Environmental impact calculations	✓✓	✓✓	✓✓	As algorithms are developed and proven over time then the resultant software modules should also be shared.
Social impact calculations	✓✓	✓✓	✓✓	As algorithms are developed and proven over time then the resultant software modules should also be shared.
Risk to life calculations	✓✓	✓✓	✓✓	As algorithms are developed and proven over time then the resultant software modules should also be shared.

Selected Requirements	Recommended level of commonality			
	Data	Algorithms	Software	Comments
Calculation defence contribution to risk	✓✓	✓✓	✓✓	More analysis required to understand if there is a benefit in sharing a software module.
Manage cost data	✓?	✓?	✓?	Commonality recommendation will require review as requirements evolve. The expectation is that there will be a single common appraisal module (for use with all tools).
Benefit-cost calculation	✓?	✓?	✓?	
MCA project appraisal calculations	✓?	✓?	✓?	

Table 3.2 shows that there are many functional requirements which can share common data and common algorithms - and much of this is already happening with RASP-related tools such as NaFRA, MDSF2 and PAMS. There is also potential to share common software modules for most of the requirement items. The benefits of sharing common software modules are strongly affected by the 'granularity' of the modules and the non-functional requirements concerning processing speed and degree of user intervention. In addition, shared modules need to be suitably designed so that the functionality required by all 'users' of the module can be provided directly (or by building on inherited features). Significant further work would be required to fully identify and specify the requirements of the appropriate common modules. From a cursory analysis of Table 3.2, the functional requirements that are the strongest candidates for development as common modules are:

- Rapid flood spreading module(s) (which use DEM data and inflow volumes to estimate flood depths). Over the next few years there are expected to be significant improvements in rapid flood spreading methods (eg from FRMRC2) these improvements and/or alternative approaches must be able to be 'plugged in' to the RASP family in the future without rewriting of the code which 'holds' the modules. Thus the initial implementation must use a generic interface not constrained by artefacts of the conceptualisation of the method.
- Economic impact calculation module
- Environmental impact module
- Social/'risk to life' impact module
- Other modules classified with two ticks in Table 3.2

There may be substantial benefits in the generation of a general probabilistic flood risk engine which combines many of the smaller core RASP modules (such as failure state analysis, overtopping calculations and defence contribution to risk). Further analysis would be required to determining the optimum 'granularity' of the shared engine components and this is recommended to facilitate the vision of shared components outlined in Figure 3.1.

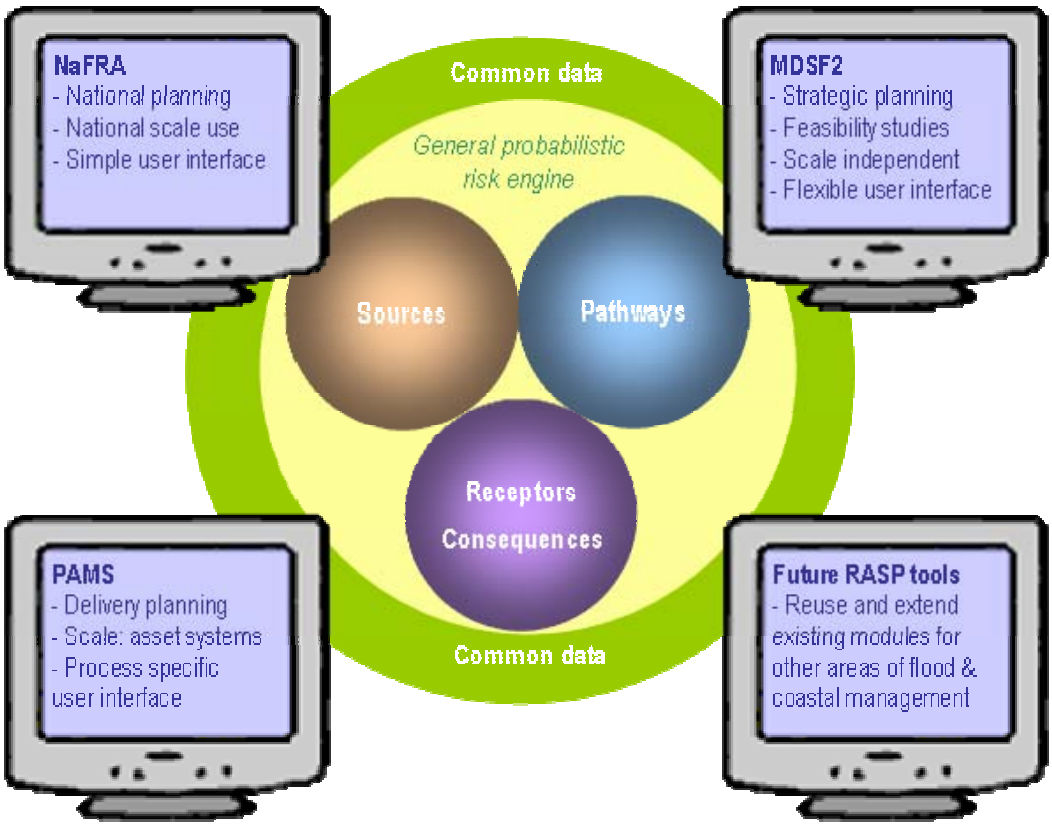


Figure 3.1 RASP family software tools sharing the RASP engine and common data

A further aspect of logical architecture is the separation of the software application into discrete layers covering:

- User interface (which provides the user interface to the application logic)
- Application logic (which is further componentised into a range of discrete modules such as those described earlier in this section)
- Data access layer (which manage the transfer of data from/to the databases)
- Databases (to store the project data)

It is recommended that RASP-related tools use the n-tier logical architecture introduced above (separate user interface, application logic, data access and database layers) as this is compliant with Environment Agency CIS requirements and will promote software sustainability.

3.3.2 Physical deployment architecture

The physical deployment architecture describes where components might be installed (i.e. on one or more items of hardware such as database servers or desktop PCs). It is considered not appropriate to make generic recommendations for the physical deployment architecture for the RASP family as this will depend on the specific requirements of the system. For example, for NaFRA the processing speed is a priority issue and therefore it is designed for

deployment across multiple processing servers, whereas MDSF2 needs to be capable to run on a single high specification PC in a consultant's office. If the n-tier logical architecture outlined above is implemented then this will facilitate a flexible physical deployment.

3.3.3 Miscellaneous Architectural/Design Aspects

Data transfer. RASP-related systems may, in the future, directly link with data sources (such as NFCDD) to provide their input data although in the short term it is expected that data links to other systems will not be dynamic (and may consist of exchange of data contained on CDs, DVDs, hard drives or over the internet). No recommendation is made here on the form of data transfer. Perhaps of more importance is to 'add value' to data by returning improved data to Agency enterprise databases following processing by RASP-related tools. Again there are a number of means for transferring this data back and no recommendations on the software aspects of this are made here.

Third-party applications. Reliance on third-party applications (such as GIS systems) should be kept to a minimum. Where links to third-party applications are appropriate then it is recommended that priority is given to applications that are already approved for use by the Agency and / or those anticipated to be in use at the commencement of / for the duration of application use (and where suitable licensing arrangements already exist).

Use of proprietary data formats. The use of proprietary data formats are discouraged as they are a hurdle to open and sustainable systems. It is recommended that XML is used wherever possible. Where use of XML would be unmanageable (e.g. large GIS datasets) then 'industry standard' formats should be used. No new proprietary data formats should be developed.

Browser-based thin client interface. It is recommended that preference is given to the use of browser-based (thin client) user interfaces for RASP-related tools. Only where the use of a browser-based interface is shown to not meet the project requirements should a rich client interface be considered. In this case it is recommended that the system is designed so that application logic can be accessed by either a browser-based interface or rich client interface (with reduced functionality for the browser-based interface).

Development language. The Agency preference is for development in Java. Development in other languages (such as C++ or C#) may be permitted if there is a strong business case. At the time of writing, there is a large code base of mature, tested RASP code already written in C# and a decision is pending on whether this can be used (probably with a Java wrapper) or whether it needs to be rewritten in Java.

Data models. The preliminary assessment of common modules described in section 3.3.1 has provided one approach at identifying component architecture. A related approach would be to undertake a data modelling exercise leading to formal definitions of a common data model for the RASP family of tools. Such a data model would seek to describe the data structure found in flood risk

analysis data sets and computations. Such a data model could provide a solid foundation for further definition of the RASP family architecture.

3.3.4 Open Systems and IPR

It is recommended that RASP-related tools are developed as open systems (a restricted open system definition is used here: a software system where the interface specifications of its components are fully defined and fully available to the Agency and its contractors).

In addition, it is recommended that source code and IPR in RASP-related tools is either owned by the Environment Agency or made freely available to the Environment Agency, its contractors and professional partners.

A further requirement is that full technical documentation is made available of the methods implemented in RASP-related tools. These requirements are necessary to help achieve the Agency objective of facilitating competition for innovation and value, and support/updating of the various applications within the RASP family. An important component of the documentation should be the use of ‘pseudo code’ to provide a structured overview for the key calculations and methods used in the code (without actually requiring someone to understand memory allocations, numeric solutions, peculiarities of individual programming languages etc.)

4 Conclusions and Recommendations

4.1 Conclusions

For the software products which are delivered by FCERM R&D projects to be readily useable by the Agency and its consultants, it is important that the software adheres to relevant software standards. The primary source of relevant standards appropriate for software to be used by the Agency is the substantial CIS document 'Enterprise Architecture: Technical Reference Model'. Guidance material, based on the Technical Reference Model, has been developed by this project and is available in the R&D Technical Report FD2121/TR2 'R&D Software Development Projects – Guidance for Research Contractors'. In addition to the objective of guiding the R&D contractor towards producing conforming software, the guidance documentation is also designed to foster early informed discussions between the R&D contractor and CIS.

The development of the FD2121 guidance has been informed by discussions with CIS and through feedback on initial drafts by selected R&D contractors. Trial applications of the guidance have demonstrated that full compliance with Technical Reference Model standards may be lead to conflicts with other project requirements. Where these conflicts occur it is important that there is dialogue between the research contractor and the Agency (CIS, Science and user representatives) and that decisions are made based on business cases and whole life costs.

The second component of the project was to review the software modularity of the RASP family of decision support tools and identify common modules and certain aspects of enabling software architecture. The RASP family of decision support tools has been described in the report 'Scoping the development and implementation of flood and coastal RASP models' (Science Report SCO50065/SR1, 2007).

The scoping report covers the background to the RASP method and makes recommendations for RASP-related research and development. It does not, however, provide sufficient detail of functional requirements to enable a definitive identification of common modules. Nevertheless, useful steps towards an appropriate architecture have been made in the present report from a review of current and ongoing RASP-related projects and through application of the FD2121 guidance. The review has shown that the tools are being designed to share common data and some common computational modules. Further action is required in the areas of defining requirements, analysing commonalities and further specification of software architecture to better achieve the objectives of facilitating the efficient production of sustainable and appropriate software tools and to facilitate competition. As requirements and methods continue to evolve it will be important to review architectural aspects and maintain an on-going dialogue with CIS and other Agency/Defra managers to facilitate take up of the software outputs.

The findings of project are compliant with the emerging Flood Risk Management Modelling Strategy (Environment Agency) and will support implementation of

the strategy by providing appropriate guidance on the Agency's preferred enterprise architecture and through supporting the modularisation of the RASP family of tools.

4.2 Recommendations

The following recommendations are made.

1. The FD2121 guidance is disseminated to those involved in ongoing FCERM R&D projects which will deliver software outputs. The potential impact of the guidance on the projects should be assessed and decisions made on whether to implement the guidance.

2. Upcoming FCERM R&D projects which will deliver software outputs should include the requirement to implement the guidance in the project specification.

3. Where appropriate, project specifications should identify the type of software that is being produced (e.g. whether the software must run on the Agency system).

4. During project initiation the FD2121 'pre contract award' assessment should be undertaken to help identify the likely involvement of CIS in the project. Resourcing for CIS involvement (e.g. strategy analyst, project architect) will need to be built into the project.

5. CIS need to clarify and streamline procedures to enable CIS documents (such as the Technical Reference Model) to be readily passed to R&D contractors.

6. The CIS requirement to use Java for R&D software development is of concern to many R&D contractors. It is recommended that further consideration is given to included .NET languages (such as C#) in the list of acceptable standards and the guidance updated.

7. General recommendations for R&D projects which deliver software include: the need to facilitate end user involvement in the development stage, the need for the project to provide complete software documentation and the need for early consideration of future custodianship, support, maintenance and user training. It would be beneficial to sharing of knowledge if the minimum contents of the documentation was standardised, for example to include well documented 'pseudo code' (i.e. provide a structured overview for the key calculations and methods used in the code without actually requiring someone to understand memory allocations, numeric solutions, peculiarities of individual programming languages etc.)

8. Further analysis is required to better define an appropriate architecture for RASP family outputs. This could be addressed through production of a formal functional and non-functional requirements definition for expected RASP-related products, combined with an initial outline

software design and a formal assessment of appropriate commonality (as outlined in Table 3.2). An alternative is to build on the modularity concepts outlined in this report, either using the source-pathway-receptor modules of a general probabilistic engine (Figure 3.1) or the ‘strong candidate modules’ listed in section 3.3.1 (including flood spreading, economic impact, environmental impact and social/risk to life impact). Currently (January 2007) the Agency is moving forward on this recommendation via the MDSF2 project (SC050051).

9. Data modelling leading to a formal common data model for the RASP family of tools should be considered as a useful component of a RASP family architecture definition. Such a data model would seek to describe the data structure found in RASP flood risk analysis data sets and computations. Newcastle University have outlined an architecture intended to be general across "computational decision support" systems, in particular tools for conducting risk analysis in the face of uncertainty. The core of this architecture consists of a data model and language, together called ‘Reframe’. Reframe is designed explicitly to facilitate transparency (and thereby also competition) and collaboration in the definition of individual modules and overall computations. It may be possible to build on elements of this work, particularly the data model, to help further develop the RASP family architecture.

Appendix A Example Applications of the Guidance

A1 Application to GLIM-CLIM Rainfall Generator

A1-3.1 General Project Information

Project Ref	
Project Title	GLIM-CLIM Rainfall Generator
Contact name	Robert Bird
Company	Halcrow Group Ltd
Tel	01793 812479
Email	BirdR@halcrow.com
Target Audience	External contractors (i.e. consultancies undertaking hydrological analysis for UK and international projects)
No. of Agency users (approx) if applicable	N/A
Project overview	
In software development terms the project will consist of some further development of existing spatial rainfall modelling software and the development of a user interface to this software. The modelling software has been developed by University College London (UCL) in Fortran and can be compiled for various operating systems (Windows, Unix, SunOS).	
Architectural Diagram	
N/A	
Other relevant information	
The development language choice for the user interface has not yet been finalised. It is likely to be either Visual Basic or C#, the former because there is existing code that could be re-used for the pre and post processing of data and the latter because the target audience will be “Windows based”, the productivity of C# and the skill base available.	

A1-3.2 Agency Software Platforms

Develop software to run harmoniously on existing Agency systems (hardware/network/software) and non-Agency systems to maximize user acceptance.

Software developed to be run on Agency systems needs to run on the platforms the Agency already uses (or will have at the time of delivery) and can support (for a synopsis of Agency platforms, correct at the time of writing see the latest, "Enterprise Architecture: Technical Reference Model").		
Response		Rationale
Will run Will not run N/A	[X] [ ] [ ]	The software will be capable of running on Agency machines, but it is envisaged the software will mainly be used by external consultants.
Software to be run on non-Agency machines should be written to maximize uptake of the software by these 3rd parties, i.e. write the software to run on the most commonly used platforms.		
Response		Rationale
Implemented Not Implemented N/A	[X] [ ] [ ]	The software will be written to work on the external consultant machines, which will mean PCs running a version of Windows (will cater for Windows 2000 and above).
Software to be run on both Agency machines and non-Agency machines should marry the requirements of the two in the best way possible. This issue must be discussed with CIS.		
Response		Rationale
Plan in place Not considered Agreed with CIS N/A	[ ] [ ] [ ] [X]	The software is not intended to be run on Agency machines.

If the timescale of development is very short the information in this document and the current "Enterprise Architecture: Technical Reference Model" can be considered up to date and should be your main reference. On larger timescale developments (over 6 months) discuss possible changing Agency platforms with CIS.

A1-3.3 Hardware Platforms

Software developed to be run on Agency machines must be developed to run on existing hardware platforms at the Agency.		
Response		Rationale
Will run Will not run N/A	[ ] [ ] [X]	Not envisaged to run on Agency machines, but could be.
Any network bandwidth usage by the software must be communicated to CIS, including details of average and burst activity (see Appendix D.4 for further information).		
Response		Rationale

Communicated Not communicated N/A	[ ] [ ] [X]	Software not to be run on Agency machines. Also, it is generally envisaged that the datasets will be held on user machines.
(Server) Processor usage should also be communicated to CIS (see Appendix D.4 for further information).		
Response		Rationale
Communicated Not communicated N/A	[ ] [ ] [X]	Processor usage will be light when using the interface generally and high when running a simulation. The processor usage will take place on the desktop PC and so will not have an adverse effect on server performance.
The need for any peripherals will need to be agreed with CIS.		
Response		Rationale
Agreed with CIS No agreement N/A	[ ] [ ] [X]	None required.

A1-3.4 Database Usage

Develop "Enterprise" database based software to run on the Agency standard database. Develop "Desktop database" software in a way that doesn't require client installs and is not locked to a proprietary format.

Database based solutions must run on the standard Agency database (currently Oracle) if the program is to be run at the Agency. Databases other than the standard enterprise database will not be allowed onto Agency systems.		
Response		Rationale
Will run Will not run N/A	[ ] [ ] [X]	This decision will be taken as part of the project. The developer currently has existing software based on Microsoft Access (DSF) which has proved useful for in-house work with the existing Fortran program. The best way forward will be investigated, which may mean that Oracle is supported or that the current DSF code can be reused. Budgetary constraints will be a key factor in this decision.
If the developed software is to be run at both Agency and non-Agency sites then if possible develop for the standard Agency database. If this is not possible or will harm uptake by the non-Agency users then write database agnostic software which will run on both the standard Agency databases and those in use by the non-Agency entities (a recommended approach in general).		
Response		Rationale
Agency standard DB Agnostic	[ ] [ ]	As above – final decision still to be taken

Other N/A	[X] [ ]	

Where software is to be developed for use at the Agency and "enterprise" databases are inappropriate for the task, desktop/embedded databases may be required. In these cases native access from within the application would be required, with no application or client installs on Agency desktop PCs. It is also required that output to a non-proprietary format (e.g. XML) is easily available from the database.

Response		Rationale
Will comply Will not comply N/A	[ ] [ ] [X]	As above – final decision still to be taken

If the software is only to run at non-Agency sites, it is still preferable that it works with the Agency standard database.

A1-3.5 Non Database Data

Do not create new proprietary data formats; store ancillary data, such as program settings, using XML file formats (see Appendix D.1 for more details).

Software developed should not write to / read from its **own** proprietary format; in general XML should be used for **new** formats. The only justification for creating proprietary formats in extreme cases might be due to performance issues, but this would have to be agreed with CIS beforehand. Where binary formats are proposed the Agency would expect to receive documentation as to the format of these and also expect some ability to handle/produce XML input/output. It is acceptable to use the de facto "standard" file formats that the Agency itself uses for things such as GIS systems.

Response		Rationale
No proprietary formats Proprietary formats N/A	[X] [ ] [ ]	It is intended that any new functionality introduced into the interface will adhere to this. The only "proprietary formats" in use are the text files used as input to the Fortran modelling program. Any formats introduced when writing the new interface software are likely to be XML based, other than if performance is an issue.

Where **ancillary** data (program settings etc.) is required to be stored you are expected to use XML as the format for this data.

Response		Rationale
XML used XML NOT used N/A	[X] [ ] [ ]	XML will be used for this.

A1-3.6 Application Architectural Compliance

Develop applications using an n-Tier, server side logic, thin client browser based approach, wherever this can satisfy the project requirements

New software developments to run on Agency machines should follow the Agency standard application architecture - an n-Tier approach, utilising a "business logic" server side in conjunction with a browser based thin client. Where this approach cannot satisfy the project requirements you will need to agree an alternate strategy with CIS, strong justification will be required (see Appendix D.6 for one possible alternative - Citrix).		
Response		Rationale
Will comply Will not comply N/A	[ ] [X] [ ]	As the software is not targeted at an internal Agency audience and given that this approach would not suit the majority of potential users, this is not the intended approach.
Software developed to run both at non-Agency sites and on Agency machines should follow the above Agency application architecture wherever possible. If this is not practical for non-Agency entities then a dual interface approach (using the same basic code base) is preferred, e.g. a rich client application at non-Agency sites and standard Agency application architecture for Agency machines.		
Response		Rationale
Agency standard Dual interface Other N/A	[ ] [ ] [ ] [X]	Not intended for Agency use

The Agency would still prefer software which will not run on Agency machines to follow the Agency application architecture, but this is not mandatory given appropriate justification.

Where updating an existing architecturally non-compliant program, the Agency would encourage a migration to a compliant state.

A1-3.7 Development Tools & Languages

Write software using Agency standard development tools.

The applicable development tool standards can be found in the latest, "Enterprise Architecture: Technical Reference Model" Agency document. In line with the application architecture requirements, the Agency standards for Enterprise scale applications is component based, n-Tier, using an application server.

When considering the development tool guidelines below, indicate where the following items are part of the justification for exceptions (research contractors should bear in mind that the overriding concerns for the Agency are **total cost**

of ownership of the software over its lifetime and whether the software satisfies **essential business requirements**):

- The program will use an existing code base written in another language (consider migration and/or making "callable" from Agency standard tools)
- Cost implications of implementing and/or developing using Agency standard tools (e.g. software license costs)
- Negative impact on functionality of software due to use of standard tools (e.g. non-interoperability with 3rd party libraries)
- Performance considerations
- Impracticality of and resistance to installation at non Agency sites
- Agency standard tools will not deliver software satisfying project requirements

One of these in isolation may not be justification for exceptions, so communicate all that apply as well as any other mitigating circumstances you believe to be relevant.

Ideally, all new development should take place in the standard development language (currently Java). Any deviation from this requires justification. If the software is not to run on Agency machines then justification will be easier.		
Response		Rationale
Will comply Will not comply N/A	[ ] [X] [ ]	It is highly unlikely we will comply in terms of using Java. The modelling software has been written by UCL and is a "finished product", tested and verified to a good academic level. The user interface may reuse elements of existing software (written in Visual Basic based) or otherwise is likely to be written in C# which can give us productivity gains, a rich Windows UI and access to a large developer skill base in the language.
Where the exact CIS standards cannot be met, you should provide justification. Note that an architecturally compliant solution (i.e. thin client browser based) is preferable to a strict adherence to specific tools.		
Response		Rationale
Standards fully met Architectural compliance Other N/A	[ ] [ ] [X] [ ]	As above.
Where the Agency standard development language cannot satisfy the project requirements, for example modelling applications, then the use of a different language could be justified – the Agency standard for modelling applications is currently C++.		
Response		Rationale

Agency standard Non standard N/A	[ ] [X] [ ]	As above.

Where a non-compliant existing application is being updated, the Agency would prefer a migration to standard development tools.

Where there is the need for some development in “legacy” languages the Agency requires that this is done in a sustainable way – as an example Fortran functionality could be “wrapped in Java” using XML, controlled by XML inputs/outputs or put into a documented DLL callable by other languages (preference in that order) to mitigate the risk to the Agency of developing in that language. If this cannot be done for any reason then a discussion with CIS will be required.

At the time of writing, the Agency policy regarding applications written for the .NET framework is to allow “commercial off the shelf” packages to be installed, but not bespoke development to be undertaken (see Appendix D.2 for more information). Justification for bespoke .NET development must address total cost of ownership issues as well as technical issues to achieve essential business requirements.

A1-3.8 Modular, Sustainable Development

Develop modular, easily extensible and reusable software to obtain maximum value from the Agency’s investment.

Contractors should develop their software in as modular a fashion as possible, using loosely coupled functions/methods probably via Object Oriented development.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	Will do this for all new code, where appropriate.
Develop software so that user interfaces are decoupled from program logic as much as possible.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	Again, will do this for all new code, where appropriate.
The use of design patterns and other modern programming techniques should be considered. Use techniques such as inheritance and encapsulation appropriately and to their best advantage.		
Response		Rationale
Done/will do Not appropriate	[X] [ ]	As above.

Not considered N/A	[ ] [ ]	
Developed software may contain functionality that itself will be useful for reuse in other software perhaps by another contractor or the Agency itself. You should make this as easy to achieve as possible and should endeavour to make it possible regardless of development environment.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	As above.
Achieve maximum interoperability by following the CIS standards along with various methods such as creating “wrappers” around software, separating code into libraries/componentisation, open communication and data exchange via SOAP and XML.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	Will do this where appropriate and where it fits in with budgetary constraints. We will look at creating a “wrapper” around the Fortran functionality.
External interfaces and available functionality should be clearly documented.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	Will do where applicable when developing the new interface.

You should consider the use of coding standards and provide details and/or references to these below.
We will use Halcrow coding standards.

A1-3.9 Security (User & Data)

Only implement application level security where absolutely necessary.

It is recommended developers refrain from implementing application level security unless absolutely necessary, especially software which is to be run on Agency systems. Any application that does this will need to be agreed with CIS. Acceptable reasons for implementing application level security are to limit access to applications where there is high processor usage which needs to be managed and where use needs to be limited to authorised users. Any securing of the system should be communicated (e.g. “the program will use standard Windows XP security”).

It should be borne in mind that a browser based application running on Agency systems is potentially accessible to all users whether valid or not - without any user level security.

If the software is to store any information pertinent to the Data Protection Act, inform CIS of what data is to be stored and how it is to be stored. This is very important. However, please note, there are other aspects of regulatory and legal compliance – guidance should be sought and will be given by the Information Management Unit (IMU) via the CIS Project/Enterprise Architect.

A recommendation when developing (server based) software is to have a management front end accessible through a browser client to perform administration.

The Agency expects that applications will not be locked down on the desktop for any reason after installation (for example, some applications require you to connect to the internet to verify the license).

It is important to explain the nature of the data that will be used or created by the program, along with how that data will be managed within the system.

Comments
It is worth noting that the rainfall data used in the simulations typically has licensing issues. The data is normally licensed to consultants by the Agency (or sometimes the Met Office) for use on a specific project only.

A1-3.10 Testing and Acceptance

Plan testing from the beginning of the project and follow the Agency testing model.

The Agency asks that contractors use the “V” model of testing (see Figure 6, “*IT Project Delivery Process (V-Model)”* below for an understanding of how this fits into the Agency development lifecycle).

Agreeing the testing and acceptance strategy will form part of the agreement when a new project is begun and should be done before any test cases are written. It is important to identify the contractual significance of acceptance within the strategy. Testing and acceptance is a conversation with CIS. Where software is to be installed on Agency PCs, the functional specification should include a test strategy. This should adopt the current Agency principles, which will be provided by the project architect.

The contractor testing process should provide evidence of the tests run and results. In particular the Agency would like to see unit and system integration tests.

Testing methods other than the ‘V’ model would need justification and to form part of the project agreement. Software updates need to go through proper pre-production testing and the testing plan should be run against them.

Where software is to be installed on Agency machines, CIS need to be given ample notice in order for them to set aside time to test the application. The complexity of the application and the type of architecture will have a bearing on the time and effort required.

The contractor is responsible for (or will be involved in) the following areas of testing:

A1-3.10.1 Unit Testing

The objective of Unit Testing is to ensure that reliable program units are produced that meet their requirements and to identify errors in program logic. Typically, the developer who coded the unit will design and run a series of tests to verify that the unit meets its requirements. Each unit should be tested individually and in isolation by exercising its inputs and observing its outputs or behaviour. There are widely used tools (unit testing frameworks) available to assist in this task such as JUnit for Java and NUnit for .NET languages.

A1-3.10.2 Integration testing

Components of code are assembled into sub-systems and linked to form a complete system. The objective is to test the relationship and links between individual units of code.

A1-3.10.3 System Testing (including FAT – Factory Acceptance Testing)

System Testing incorporates both functional and non functional testing. Functional system testing is focused on testing the system based on what it is supposed to do as defined in the functional requirements. Non functional system testing looks at those aspects that are important yet not directly related to what functions the system performs. Non-functional requirements are just as important as functional requirements and in all cases it is vital that these are tested prior to the launch of the system. It includes aspects like performance and security which are considered vital for today’s web based applications.

A1-3.10.3 System Test Process (involvement by 3rd Party Suppliers)

To ensure quality, prior to delivery to the Agency, the 3rd Party supplier will be required to demonstrate compliance by hosting Acceptance Testing on their site, this will in effect be a pre-User Acceptance Test known by the Agency as Factory Acceptance Testing (FAT). This process will have defined, agreed acceptance criteria and will be subject to a Test Readiness Review Meeting prior to deployment on any Agency infrastructure. In addition, the 3rd party supplier will assist in the installation of the new application on the agency site.

This will be done as part of the Acceptance Testing stage and will again require the 3rd party supplier to show compliance on the Agency's infrastructure.

A1-3.10.4 Site Acceptance Testing (SAT)

SAT will be used as a quality check to ensure that when the application is installed onto the Agency infrastructure that it functions correctly with no critical errors. The 3rd Party supplier will be on site to assist and ensure that the application, specifically the server code (where applicable), can be deployed correctly. The 3rd Party supplier will then conduct a subset of their System Tests to prove that the application can function without any critical errors. If possible, at this stage the opportunity should be taken for the 3rd party to forward any test assets such as test scripts that may be re-usable by the Agency.

A1-3.10.5 User Acceptance Testing (UAT)

UAT will take place on the Agency infrastructure and will confirm that the system meets its business requirements.

When dealing with 3rd parties, the Test Management Process should also be carefully considered. The Agency Test Manager/Coordinator will decide if the project should produce a combined Test Strategy taking into account all testing stages or if separate strategies should be produced, one by the Agency and one by the 3rd party. This will also apply to Test plans and approach documents.

Comments
As the software is not intended for installation at the Agency, it is expected the developer will follow its own testing procedures in the development of the software (much of which ties in with the Agency guidelines above).

A1-4 Implementation Planning

Follow the Agency deployment procedures, determine who is responsible for support and enable a smooth implementation with no "nasty surprises".

A1-4.1 Software Deployment

For software to be installed on Agency machines, the contractor must follow the Agency standards for software deployment/install – see Appendix C for more detail.

Suppliers of proposed systems must provide documented support for the application integration task to the standardised Agency desktop.

Where desktop software is to be installed on standard Agency desktops, CIS expect the contractor to detail the impact on the workstation, including the following:

- Assurance that the installation/application makes no changes to or deletions of protected operating system files.
- A list of dependent components (e.g. Active X controls, DLLs, drivers etc.)
- A list of Dynamic Link Library (DLL) and Application Programming Interface (API) calls.
- A list of known changes to registry keys

Comments
As the software is not planned for Agency installation we will not generally be releasing this information to consultants as a matter of course, but can provide it on request.

A1-4.2 Transition to Support and Maintenance

Before implementation, the decision needs to be made as to who performs the support function – the supplier, CIS or both. In general software which is not to be run on Agency machines will be the responsibility of the original contractor.

The service level required for the support must be set as must any need for business lead users, database administrators etc. Further to this, developers will need to complete the Agency document: “Service Support Requirements Brief” for projects that are to be handed over to the Agency. It is preferred this takes place as early as possible so that CIS can plan adequately ahead and spot any potential problems. Once the software goes into production, the document contents will generally be translated into a ‘Service Level Agreement’.

Specific supporting documents may need to be provided (such as Entity Relationship diagrams) as agreed with the Agency Project Manager.

Code storage and version control of the software is the responsibility of the contractor. However, where the intellectual property of the software belongs to the Agency then the source code of distributed production versions should be submitted to the Agency library.

Who will support the application?		
Response		Rationale
Agency CIS Contractor Other Not yet known	[ ] [X] [ ] [ ]	Likely to be sold as a COTS (“commercial off the shelf”) program.

A1-4.4 Storage Requirements

As part of planning for implementation on Agency machines, physical storage requirements must be evaluated and communicated to CIS. The principal areas to consider are:

- Frequency of backup – how often the program data needs to be backed up
- Recovery time – the speed of turnaround required if the program/data needs to be restored from backup
- Amount of storage required – how much storage the program and data will require now and how much will it grow in the future.

Storage requirements	
Frequency of backup required	Will depend on frequency of use of program
Recovery time	Will depend on frequency of use of program
Amount of storage required (now)	Measured in GB, allow 2GB min per project
Amount of storage required (future)	Measured in GB, allow 2GB min per project

A2 Application to MDSF2

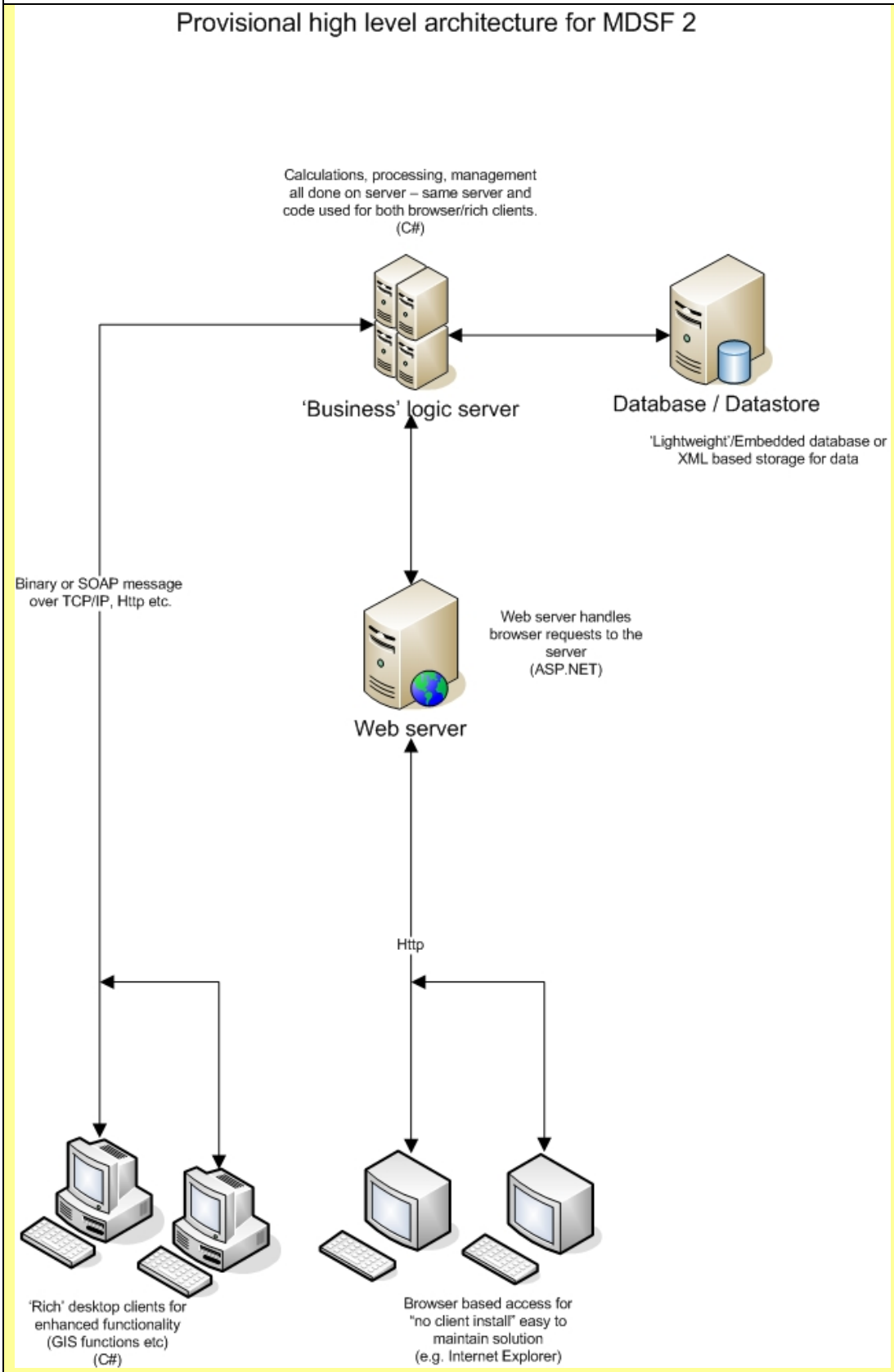
Note that standard text from the Guidance document has been deleted from the following example.

A2-3.1 General Project Information

Project Ref	MDSF 2
Project Title	Modelling Decision Support Framework 2 (MDSF2)
Contact name	Jon Wicks
Company	Halcrow Group Limited
Tel	01793 812479
Email	wicksjm@halcrow.com
Target Audience	Agency staff concerned primarily with CFMPs, SMPs, strategy and scheme studies and the consultants engaged upon these tasks on behalf of the Agency.
No. of Agency users (approx) if applicable	75 requiring GIS functionality etc., running processes (estimate) 150 "viewers" – viewing output from processes (estimate)
Project overview	
The overall objective of the project is to extend and improve the existing, first version of MDSF (MDSF1) to incorporate new and improved risk-based methods in order to provide a better and more consistent decision support tool for both the CFMP and SMP programmes, strategies and scheme appraisal. The Modelling and Decision Support Framework (MDSF) was developed in 2001 to provide a tool for quantifying economic and social impacts of flooding at catchment scale for present day conditions, future scenarios and with flood management options. The present version of MDSF, however, uses only a simplified representation of the role of defences and does not properly take account of defence performance in the analysis of risks and their management. This is a particularly crucial point in the context of understanding and managing the actual risk. MDSF2 will incorporate the RASP (Risk Assessment for Strategic Planning) approach that has been developed to take into account the performance of flood defences. The project will also address a number of software issues such as GIS platform which have been obstacles to widespread uptake within the Agency. **Main objectives** To improve the present version of MDSF by incorporating an appropriate level of the RASP methodology to allow MDSF to assess the performance of defences better and thus support a full range of catchment, estuary and coastal flood planning and option appraisal tasks in an efficient, consistent and transparent way.	

Build upon the present MDSF and the work of the RASP methods to produce an item of software under an approved QA system which can be efficiently used by operating authorities and their consultants. To put in links to other strategic systems and projects such as NFCDD, Flood Mapping Programme and PAMS, and to consider future links to similar systems in land and water quality. To facilitate the inclusion (but not to include under this phase) the option appraisal of non-structural options such as rural and urban land management, flood event management and flood loss management thus laying a foundation for a tool which can support the Agency's policy of integrated flood risk management**.** To ensure that software development is as far as possible 'future-proofed' by reducing to a realistic minimum its dependence on specific third party software; and to ensure that the software is modular, so that individual modules of MDSF2 can be re-used in other applications in the RASP family and vice versa.

Architectural Diagram



Other relevant information
The first version of MDSF is not scalable to the requirements for MDSF 2 and as such large parts of the software will be written from scratch, although a significant existing (RASP) code base provides a good starting point for risk based elements. MDSF1 will act as a prototype and lessons learned will be incorporated into MDSF2.

A2-3.2 Agency Software Platforms

Develop software to run harmoniously on existing Agency systems (hardware/network/software) and non-Agency systems to maximize user acceptance.

Software developed to be run on Agency systems needs to run on the platforms the Agency already uses (or will have at the time of delivery) and can support (for a synopsis of Agency platforms, correct at the time of writing see the latest, "Enterprise Architecture: Technical Reference Model").		
Response		Rationale
Will run Will not run N/A	[X] [ ] [ ]	Software will be designed to run on current Agency platforms.
Software to be run on non-Agency machines should be written to maximize uptake of the software by these 3rd parties, i.e. write the software to run on the most commonly used platforms.		
Response		Rationale
Implemented Not Implemented N/A	[X] [ ] [ ]	We will write the software to best fit with the target audience (primarily Windows 2000/XP).
Software to be run on both Agency machines and non-Agency machines should marry the requirements of the two in the best way possible. This issue must be discussed with CIS.		
Response		Rationale
Plan in place Not considered Agreed with CIS N/A	[X] [ ] [ ] [ ]	Still to be agreed with CIS. Our proposed solution is designed to maximise take-up of the software by non Agency entities, one of the key project aims. Other sections of the document explain some of the steps taken to achieve this, e.g. a database agnostic approach

A2-3.3 Hardware Platforms

Software developed to be run on Agency machines must be developed to run on existing hardware platforms at the Agency.	
Response	Rationale

Will run Will not run N/A	[X] [ ] [ ]	Will be designed from the beginning to do so.
Any network bandwidth usage by the software must be communicated to CIS, including details of average and burst activity (see Appendix D.4 for further information).		
Response		Rationale
Communicated Not communicated N/A	[ ] [X] [ ]	Full details not known at this stage. It is expected at this stage that general network traffic will be quite low, whilst there are 100MB+ files that will need to be accessed and processed at times.
(Server) Processor usage should also be communicated to CIS (see Appendix D.4 for further information).		
Response		Rationale
Communicated Not communicated N/A	[ ] [X] [ ]	Unknown at this stage. However, there will be periods of high processor usage – long (e.g. hour long runs) calculations. The various deployment options will allow for the main processing to take place on server machines if required (not recommended due to cost of providing burst CPU) or on client PCs (the advantage being that this is inherently self scaling). It is not a core requirement of the project, but we would hope to be able to use distributed processing for long runs (e.g. using Condor), with the system “degrading gracefully” if clustering is not available.
The need for any peripherals will need to be agreed with CIS.		
Response		Rationale
Agreed with CIS No agreement N/A	[ ] [ ] [X]	None required.

A2-3.4 Database Usage

Develop “Enterprise” database based software to run on the Agency standard database. Develop “Desktop database” software in a way that doesn’t require client installs and is not locked to a proprietary format.

Database based solutions must run on the standard Agency database (currently Oracle) if the program is to be run at the Agency. Databases other than the standard enterprise database will not be allowed onto Agency systems.		
Response		Rationale
Will run Will not run	[X] [ ]	Will be designed from the beginning to work with Oracle 9i v2 and also the current version (10).

N/A	[ ]	

If the developed software is to be run at both Agency and non-Agency sites then if possible develop for the standard Agency database. If this is not possible or will harm uptake by the non-Agency users then write database agnostic software which will run on both the standard Agency databases and those in use by the non-Agency entities (a recommended approach in general).

Response		Rationale
Agency standard DB Agnostic Other N/A	[ ] [X] [ ] [ ]	Software is to run at non-Agency sites and Oracle cannot be enforced as the database of choice at these sites. To maximise uptake it is proposed that SQL Server (Full + MSDE) is supported at a minimum as well as Oracle.

Where software is to be developed for use at the Agency and “enterprise” databases are inappropriate for the task, desktop/embedded databases may be required. In these cases native access from within the application would be required, with no application or client installs on Agency desktop PCs. It is also required that output to a non-proprietary format (e.g. XML) is easily available from the database.

Response		Rationale
Will comply Will not comply N/A	[X] [ ] [ ]	Although MDSF2 will run ‘standalone’ using an optionally installed database engine this will not be required for Agency deployment.

If the software is only to run at non-Agency sites, it is still preferable that it works with the Agency standard database.

A2-3.5 Non Database Data

Do not create new proprietary data formats; store ancillary data, such as program settings, using XML file formats (see Appendix D.1 for more details).

Software developed should not write to / read from its **own** proprietary format; in general XML should be used for **new** formats. The only justification for creating proprietary formats in extreme cases might be due to performance issues, but this would have to be agreed with CIS beforehand. Where binary formats are proposed the Agency would expect to receive documentation as to the format of these and also expect some ability to handle/produce XML input/output. It is acceptable to use the de facto “standard” file formats that the Agency itself uses for things such as GIS systems.

Response		Rationale
No proprietary formats Proprietary formats N/A	[X] [ ] [ ]	We will adopt XML as our standard format for storing MDSF2-generated data such as workflow descriptions. Industry standard native formats (such as ESRI GIS formats) will be used where necessary.

Where **ancillary** data (program settings etc.) is required to be stored you are expected to use XML as the format for this data.

Response		Rationale
XML used XML NOT used N/A	[X] [ ] [ ]	MDSF2-generated ancillary data such as program settings will be stored using XML.

A2-3.6 Application Architectural Compliance

Develop applications using an n-Tier, server side logic, thin client browser based approach, wherever this can satisfy the project requirements

New software developments to run on Agency machines should follow the Agency standard application architecture - an n-Tier approach, utilising a "business logic" server side in conjunction with a browser based thin client. Where this approach cannot satisfy the project requirements you will need to agree an alternate strategy with CIS, strong justification will be required (see Appendix D.6 for one possible alternative - Citrix).		
Response		Rationale
Will comply Will not comply N/A	[ ] [X] [ ]	Unfortunately, it is anticipated that these standards would severely harm the project. A browser based approach will not be able to deliver the integration with GIS tools that is required by the program. We are however planning to develop an n-Tier logical architecture with a variety of different deployment options. The deployment would be flexible. It could consist of a data access layer on one machine, the actual database on another, the "business logic" on another and the client on another. At the other extreme it would be capable of deployment onto a single machine at its most simple level. This would satisfy the need for both corporate, distributed style deployment and single standalone users.

Software developed to run both at non-Agency sites and on Agency machines should follow the above Agency application architecture wherever possible. If this is not practical for non-Agency entities then a dual interface approach (using the same basic code base) is preferred, e.g. a rich client application at non-Agency sites and standard Agency application architecture for Agency machines.

Response		Rationale
Agency standard Dual interface Other	[ ] [ ] [X]	This has been considered, but cannot be justified primarily due to cost issues and time constraints. A dual interface is technically feasible due to the

N/A	[ ]	adoption of a clear n-tier logical architecture, but as stated earlier, the browser based approach is unlikely to provide the level of interaction with GIS datasets required, with response times likely to be far too sluggish.

A2-3.7 Development Tools & Languages

Write software using Agency standard development tools.

Ideally, all new development should take place in the standard development language (currently Java). Any deviation from this requires justification. If the software is not to run on Agency machines then justification will be easier.		
Response		Rationale
Will comply Will not comply N/A	[ ] [X] [ ]	There is already a large code base of mature, tested, code (RASP) which is written in C# and is intended to be re-used (no budget to rewrite). Development is planned to be in C# (using the .NET framework). Through the .NET framework and its managed memory model (as with Java), many performance, reliability and security issues are handled internally. The use of C# will also make best use of the skills and tools existing at the research contactors. A further contributing issue is that C# skills are widely available amongst potential research contractors whereas Java skills are less widely available.
Where the exact CIS standards cannot be met, you should provide justification. Note that an architecturally compliant solution (i.e. thin client browser based) is preferable to a strict adherence to specific tools.		
Response		Rationale
Standards fully met Architectural compliance Other N/A	[ ] [ ] [X] [ ]	We will be “architecturally compliant” in terms of an n-Tier logical architecture, separating out the business logic and decoupling it from the user interface etc. We won’t however be fully compliant due to not developing as a browser based application. We will not be able to make use of an application server such as Weblogic as we need to maximise the uptake of the software at non Agency sites and cannot expect them to install this due to cost/administration/policy issues. If a web user interface were to be developed at a later stage, it could sit within such an application server.

Where the Agency standard development language cannot satisfy the project requirements, for example modelling applications, then the use of a different language could be justified – the Agency standard for modelling applications is currently C++.		
Response		Rationale
Agency standard Non standard N/A	[ ] [X] [ ]	As above. Also users (at non Agency sites in particular) will expect a familiar, “native” Windows application. Also, the C# language is similar to Java and arguably more readable for a Java developer than C++.

A2-3.8 Modular, Sustainable Development

Develop modular, easily extensible and reusable software to obtain maximum value from the Agency’s investment.

Contractors should develop their software in as modular a fashion as possible, using loosely coupled functions/methods probably via Object Oriented development.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	The software will be designed in this way. In addition it is planned that software components will be “pluggable”, so that modifications to sub components will not require core application changes. New functionality can be added without opening the core code.
Develop software so that user interfaces are decoupled from program logic as much as possible.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	This is an integral part of the overall architecture of the program and the planned “pluggable” architecture and XML driven workflow will help deliver this.
The use of design patterns and other modern programming techniques should be considered. Use techniques such as inheritance and encapsulation appropriately and to their best advantage.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	Design patterns in particular will be used if and where applicable, in particular the ‘Gang of Four’ patterns. Other modern techniques to be used include unit testing, and a RUP based process with UML diagramming.

Developed software may contain functionality that itself will be useful for reuse in other software perhaps by another contractor or the Agency itself. You should make this as easy to achieve as possible and should endeavour to make it possible regardless of development environment.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	This will be partially achieved by the separation of application logic from the user interface. There are also currently outline plans for a “workflow” design which would allow developers/power users to “string” together the blocks of functionality to produce a new set of processes.
Achieve maximum interoperability by following the CIS standards along with various methods such as creating “wrappers” around software, separating code into libraries/componentisation, open communication and data exchange via SOAP and XML.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	Data exchange between modules is currently planned to take place by both memory (for speed) and XML for interoperability. Where required for deployment, components will present SOAP interfaces. Large datasets (such as GIS data) will generally be communicated in their native formats to conserve bandwidth (due to XML ‘bloat’ etc.).
External interfaces and available functionality should be clearly documented.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	This will be done as a matter of course (through both separate documentation and internal code documentation tools such as XMLDoc/JavaDoc as appropriate).

You should consider the use of coding standards and provide details and/or references to these below.
We will use Halcrow coding standards.

A2-3.9 Security (User & Data)

Only implement application level security where absolutely necessary.

Comments
There will be no data storage requirements pertinent to the Data Protection Act. It is unlikely that security will need to be implemented beyond restricted access to associated network storage locations and databases to Agency users. Access to the system will be via existing rights management systems, i.e. we will not create our own – access will be secured using standard Windows security.

Some data may have licensing restrictions and MDSF2 will support the recording of such information. This will generally take the form of the Agency granting licenses to use its data to external consultants.

A2-3.10 Testing and Acceptance

Plan testing from the beginning of the project and follow the Agency testing model.

Comments
It is currently planned to follow the testing guidelines fully.

A2-4 Implementation Planning

Follow the Agency deployment procedures, determine who is responsible for support and enable a smooth implementation with no "nasty surprises".

A2-4.1 Software Deployment

For software to be installed on Agency machines, the contractor must follow the Agency standards for software deployment/install – see Appendix C for more detail.

Suppliers of proposed systems must provide documented support for the application integration task to the standardised Agency desktop.

Comments
We will supply all the required information. We currently plan that the install will be an XCopy type installation (facilitated by the .NET framework) – that is, a matter of copying the program files to a folder on the user's machine – easily accomplished with the Agency standard deployment tools. This obviously depends on the prerequisite that the .NET framework is installed (which can also be automated with the Agency standard deployment tools).

A2-4.2 Transition to Support and Maintenance

Who will support the application?		
Response		Rationale
Agency CIS Contractor Other Not yet known	[ ] [ ] [ ] [X]	It is probable that Agency CIS will provide first line support to their users and an agreement on this and support of non Agency users will be needed. An SLA will be required to cover non Agency use and 2nd line support to Agency CIS.

A2-4.3 Storage Requirements

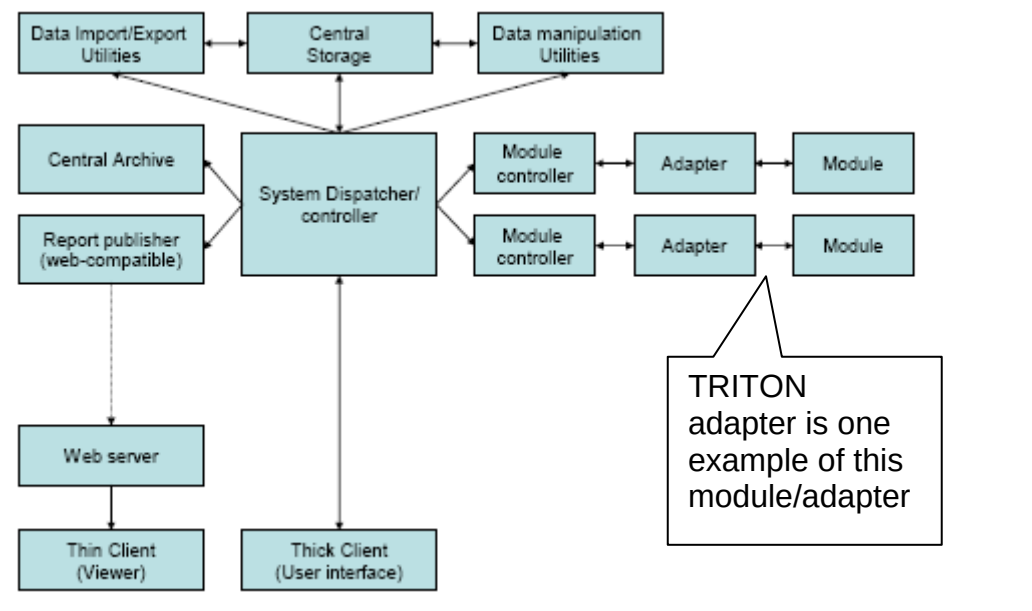
Storage requirements	
Frequency of backup required	Weekly (Backup should be to Agency standards for project related data).
Recovery time	Same day (Recovery should be to Agency standards for project related data).
Amount of storage required (now)	Unknown, very dependent on no. of users and datasets used. However, the amount of storage required will be very significant if the Agency decides to store the data returned by consultants on the live system (this is not done with the current MDSF1 system)
Amount of storage required (future)	Unknown, very dependent on no. of users and datasets used, but will grow. Again it is also dependent on what is done with the data returned by consultants.

A3 Application to NFFS (TRITON Model Adapter)

Note that standard text from the Guidance document has been deleted from the following example.

A3-3.1 General Project Information

Project Ref	NFFS
Project Title	TRITON Module Adapter
Contact name	n/a
Company	n/a
Tel	
Email	
Target Audience	Agency staff using the TRITON model within the NFFS system
No. of Agency users (approx) if applicable	Forecasting staff in many regions
Project overview	
For background information on the NFFS and the TRITON module see Appendix B. The TRITON module, which is addressed in this case study, started life as a fully interactive stand-alone program running on a number of PCs in various Agency regions. It was developed long before the idea of NFFS evolved so when the requirement to implement TRITON as a module in NFFS manifested itself, there was a considerable amount of work involved to convert the software into something which could be integrated into the NFFS. The software needed to be upgraded to comply with the NFFS requirements as summarized in Appendix B.	

Architectural Diagram

Other relevant information

A3-3.2 Agency Software Platforms

Develop software to run harmoniously on existing Agency systems (hardware/network/software) and non-Agency systems to maximize user acceptance.

Software developed to be run on Agency systems needs to run on the platforms the Agency already uses (or will have at the time of delivery) and can support (for a synopsis of Agency platforms, correct at the time of writing see the latest, "Enterprise Architecture: Technical Reference Model").		
Response		Rationale
Will run Will not run N/A	[X] [ ] [ ]	Needs to run on the NFFS Forecasting Shell Servers
Software to be run on non-Agency machines should be written to maximize uptake of the software by these 3rd parties, i.e. write the software to run on the most commonly used platforms.		
Response		Rationale
Implemented Not Implemented N/A	[ ] [ ] [X]	

Software to be run on both Agency machines and non-Agency machines should marry the requirements of the two in the best way possible. This issue must be discussed with CIS.		
Response		Rationale
Plan in place Not considered Agreed with CIS N/A	[ ] [ ] [ ] [X]	

A3-3.3 Hardware Platforms

Software developed to be run on Agency machines must be developed to run on existing hardware platforms at the Agency.		
Response		Rationale
Will run Will not run N/A	[X] [ ] [ ]	Needs to run on the NFFS Forecasting Shell Servers Available disk space is not an issue on these systems. A module would not expect to store e.g. historic forecasts on the server, for example. These are stored within NFFS.
Any network bandwidth usage by the software must be communicated to CIS, including details of average and burst activity (see Appendix D.4 for further information).		
Response		Rationale
Communicated Not communicated N/A	[ ] [X] [ ]	The modular approach is such that NFFS delivers all the raw data the module needs, the module manipulates this data and produces forecasts which are finally retrieved by NFFS which also tidies up the inputs and outputs by removing them from the server. The result is that there are two bursts of network use which coincide with the delivery of module inputs and the retrieval of the module outputs. NFFS will be configured to only deliver exactly what the module needs, thereby avoiding unnecessary traffic. It is the responsibility of the module to ensure that it only produces the outputs which NFFS requires. Note that the use of binary data transfer, which is described in Appendix B, can dramatically reduce the amount of data being transferred, whilst still adhering to the NFFS standards. The module should not directly cause any network traffic to be generated and it should only access module data which is local to the directory where the module is installed. It should have no need whatsoever to communicate directly with "the outside world".
(Server) Processor usage should also be communicated to CIS (see Appendix D.4 for further information).		
Response		Rationale

Communicated Not communicated N/A	[ ] [X] [ ]	As with the network traffic, the module will be called on a one-off basis by NFFS, usually under a schedule but occasionally because of a NFFS user request. In both cases, the module is invoked by NFFS and runs to completion and must exit in a controlled fashion. The amount of CPU use depends on how much processing needs to be done during this run but it is important to stress that all modules which are integrated into NFFS are normally inactive and non-interactive, unlike a stand-alone R&D application which would most likely be running all the time, possibly waiting for user input via a keyboard or mouse. Modules cannot be left running as NFFS waits for them to complete and then uses completion as a signal to fetch the results produced.

The need for any peripherals will need to be agreed with CIS.

Response		Rationale
Agreed with CIS No agreement N/A	[ ] [ ] [X]	There should be no need for an NFFS module to interface to any peripherals as all forecasts produced by the module are returned to NFFS which is the platform from which users can view, manipulate and output forecasts as necessary.

A3-3.4 Database Usage

Develop "Enterprise" database based software to run on the Agency standard database. Develop "Desktop database" software in a way that doesn't require client installs and is not locked to a proprietary format.

Database based solutions must run on the standard Agency database (currently Oracle) if the program is to be run at the Agency. Databases other than the standard enterprise database will not be allowed onto Agency systems.		
Response		Rationale
Will run Will not run N/A	[ ] [ ] [X]	A NFFS module should not be a database based solution for the simple reason that all the data it needs is delivered to it by NFFS and all the outputs it produces are retrieved by NFFS. There should never be a need for a module to store historic input or output data. Even if a module uses the results of a previous forecast to initialise the next one, it is not correct practice to store the previous forecast on the local Forecasting Shell Server. Instead NFFS will maintain a "database" of previous forecasts, stored as BLOBS, and can be configured to deliver exactly what the module needs.

		Most NFFS modules will have a local Module data set which comprises static information. In the case of TRITON, for example, this includes matrices which are complex look-up tables. This data is in a format which is bespoke / convenient to the module and does not need to conform to NFFS standards. There is no reason why a local database (e.g. Oracle) could not be used to store module data if this is necessary and preferred. The main criteria should be that as the module data set will need to be maintained in the event that changes are necessary, it is important that the changes can be easily made, the module data set file formats are well documented and that any off-line programs which are used to maintain this local data are also documented and made available to the Agency as part of the module.

If the developed software is to be run at both Agency and non-Agency sites then if possible develop for the standard Agency database. If this is not possible or will harm uptake by the non-Agency users then write database agnostic software which will run on both the standard Agency databases and those in use by the non-Agency entities (a recommended approach in general).

Response		Rationale
Agency standard DB Agnostic Other N/A	[ ] [ ] [ ] [X]	See response above

Where software is to be developed for use at the Agency and “enterprise” databases are inappropriate for the task, desktop/embedded databases may be required. In these cases native access from within the application would be required, with no application or client installs on Agency desktop PCs. It is also required that output to a non-proprietary format (e.g. XML) is easily available from the database.

Response		Rationale
Will comply Will not comply N/A	[ ] [ ] [X]	See response above

A3-3.5 Non Database Data

Do not create new proprietary data formats; store ancillary data, such as program settings, using XML file formats (see Appendix D.1 for more details).

Software developed should not write to / read from its **own** proprietary format; in general XML should be used for **new** formats. The only justification for creating proprietary formats in extreme cases might be due to performance issues, but this would have to be agreed with CIS beforehand. Where binary formats are proposed the Agency would expect to receive documentation as to the format of these and also expect some ability to handle/produce XML input/output. It is acceptable to use the de facto "standard" file formats that the Agency itself uses for things such as GIS systems.		
Response		Rationale
No proprietary formats Proprietary formats N/A	[X] [X] [ ]	TRITON will continue to use own data formats but NFFS PI XML will be used for all data that needs to be transferred between NFFS and TRITON
Where **ancillary** data (program settings etc.) is required to be stored you are expected to use XML as the format for this data.		
Response		Rationale
XML used XML NOT used N/A	[X] [ ] [ ]	XML used for transfer of data between TRITON and NFFS

A3-3.6 Application Architectural Compliance

Develop applications using an n-Tier, server side logic, thin client browser based approach, wherever this can satisfy the project requirements

New software developments to run on Agency machines should follow the Agency standard application architecture - an n-Tier approach, utilising a "business logic" server side in conjunction with a browser based thin client. Where this approach cannot satisfy the project requirements you will need to agree an alternate strategy with CIS, strong justification will be required (see Appendix D.6 for one possible alternative - Citrix).		
Response		Rationale
Will comply Will not comply N/A	[ ] [ ] [X]	This is not really relevant for NFFS modules as the modules are only one component (part of the business logic) of the larger n-Tier NFFS system. The NFFS architecture can loosely be described as "slim-client" in that it minimises the band-width requirement by effectively running processes on distributed systems, making fully use of their processing power. The main tasks carried out by NFFS in running a module are pushing and pulling data from these distributed systems.
Software developed to run both at non-Agency sites and on Agency machines should follow the above Agency application architecture wherever possible. If this is not practical for non-Agency entities then a dual interface approach (using the same basic code base) is preferred, e.g. a rich client application at non-Agency sites and standard Agency application architecture for Agency		

machines.		
Response		Rationale
Agency standard Dual interface Other N/A	[ ] [ ] [ ] [X]	See above

A3-3.7 Development Tools & Languages

Write software using Agency standard development tools.

Ideally, all new development should take place in the standard development language (currently Java). Any deviation from this requires justification. If the software is not to run on Agency machines then justification will be easier.		
Response		Rationale
Will comply Will not comply N/A	[ ] [ ] [ ]	*Information not available*
Where the exact CIS standards cannot be met, you should provide justification. Note that an architecturally compliant solution (i.e. thin client browser based) is preferable to a strict adherence to specific tools.		
Response		Rationale
Standards fully met Architectural compliance Other N/A	[ ] [ ] [ ] [ ]	*Information not available*
Where the Agency standard development language cannot satisfy the project requirements, for example modelling applications, then the use of a different language could be justified – the Agency standard for modelling applications is currently C++.		
Response		Rationale
Agency standard Non standard N/A	[ ] [ ] [ ]	*Information not available*

A3-3.8 Modular, Sustainable Development

Develop modular, easily extensible and reusable software to obtain maximum value from the Agency's investment.

Contractors should develop their software in as modular a fashion as possible, using loosely coupled functions/methods probably via Object Oriented development.

Response		Rationale
Done/will do Not appropriate Not considered N/A	[ ] [ ] [ ] [ ]	*Information not available*

Develop software so that user interfaces are decoupled from program logic as much as possible.

Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	All NFFS modules are "black-box" and as such have no user interaction whatsoever. They reside on servers which are remote and are only run when called by NFFS. Users do not normally have access to the servers and should not even need to know where the module is installed. They should ideally be produced as console applications which have no graphical interface, as this is an overhead which can be avoided.

The use of design patterns and other modern programming techniques should be considered. Use techniques such as inheritance and encapsulation appropriately and to their best advantage.

Response		Rationale
Done/will do Not appropriate Not considered N/A	[ ] [ ] [ ] [ ]	*Information not available*

Developed software may contain functionality that itself will be useful for reuse in other software perhaps by another contractor or the Agency itself. You should make this as easy to achieve as possible and should endeavour to make it possible regardless of development environment.

Response		Rationale
Done/will do Not appropriate Not considered N/A	[ ] [ ] [ ] [ ]	*Information not available*

Achieve maximum interoperability by following the CIS standards along with various methods such as creating “wrappers” around software, separating code into libraries/componentisation, open communication and data exchange via SOAP and XML.

Response		Rationale
Done/will do Not appropriate Not considered N/A	[ ] [ ] [ ] [ ]	*Information not available*

External interfaces and available functionality should be clearly documented.		
Response		Rationale
Done/will do Not appropriate Not considered N/A	[X] [ ] [ ] [ ]	Modules will not need to communicate with external systems to (e.g.) poll for data via telemetry or access files on other networked servers. They must be self-contained and only need to use data which is provided to them by NFFS.

You should consider the use of coding standards and provide details and/or references to these below.
Information not available

A3-3.9 Security (User & Data)

Only implement application level security where absolutely necessary.

Comments
There will be no data storage requirements pertinent to the Data Protection Act. The NFFS Modules, as the executable code, resides on systems which are totally inaccessible by "normal" users. Likewise, NFFS modules will not be browser based. The "management front end" is actually in this case NFFS as only authorised users can access NFFS and in turn the supported modules. NFFS keeps a comprehensive history of use and any un-authorised running of modules would be noted and could be diagnosed. Modules should not have any restrictions as to who can activate them, when or how often they are run. NFFS in fact becomes the single "user" of the module within the Agency. Any licence issues need to be addressed during procurement.

Appendix B NFFS Background Information

B1 NFFS High Level Solution Architecture

B1.1 NFFS Logical Model

The high level user requirements (use cases) were analysed and as a result, a logical model of the system components has been produced. This is illustrated below in Figure B1.1.

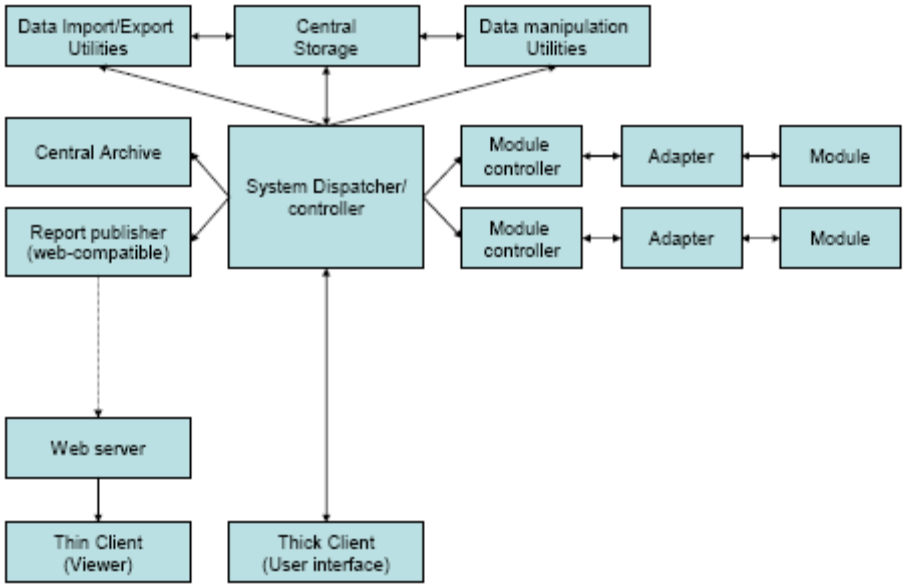


Figure B1.1 NFFS Client Server Logical Design

Figure B1.1 does not prescribe a particular architecture (i.e. the above could all exist on a single or multiple physical machines). Note that two module controllers and associated adapters/modules are shown. In fact, there are likely to be more than this, subject to the constraints of the modules and the hardware infrastructure. In addition, the above does not preclude the use of a single system for all regions or independent systems for each Agency region. For a standalone system the following configuration would be appropriate (Figure B1.2):

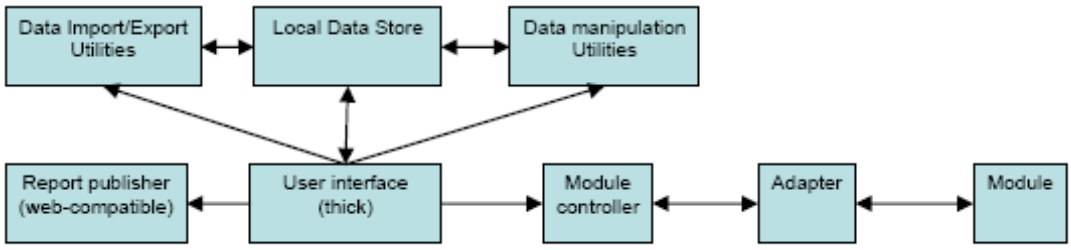


Figure B1.2. NFFS Standalone Configuration

The logical components are briefly described below.

- **System Controller/Dispatcher:** This component is responsible for scheduling and dispatching requests to execute module runs and other

defined tasks. It maintains feedback on the status of the system components and any active module runs in progress.

- **Central Storage;** This facility provides storage for module datasets, boundary conditions and output (including predefined reports). This storage facility has expiration conditions. A separate section is incorporated to provide archiving facility for critical module runs (Module datasets, boundary conditions, output (including predefined reports), executables, logbook).

- **Data Manipulation Utilities:** Utilities to manipulate the large raw module output files and to generate “slices” and subsets for analysis and display.

- **Data Import/Export Utilities:** Utility to allow the import and export of data files.

- **Module Controller:** Component to manage the execution and monitoring of the module. Usually the Modules are instantiated via the controller using a predefined script or batch file.

- **Adapter:** Provides a standard published interface between the NFFS and third party modules. This enables the system to provide the data in the required format and location that is expected by the third party modules.

- **Module:** The executable(s) encapsulating the modelling physics (e.g. ISIS or TRITON).

- **User Interface (thick):** Provides the interface through which the user can:
    - Submit requests for module runs.
    - Selecting and modify input datasets
    - Select and display subsets of module output etc.

- **User Interface (thin):** Provides the user interface to browse data sets published on the web server

- **Report Publisher (web compatible):** Enables generation of reports according to predefined layout formats in web compatible file formats (HTML, JPEG, GIF, PNG)[1]

B1.2 NFFS System Architecture

The NFFS system is based upon a number of independent Java based clients and processes. Communication and data transfer between the central system and the “shell servers” as well as the (thick) clients, is based upon the Java Messaging Service (JMS).

The key system entities, the central system, the Shell servers (the numerical engines) and thick clients, are all decoupled from each other. This ensures independent and concurrent operation as well as flexibility in selecting their location (both in terms of servers and sites).

It allows the system to make appropriate use of available network bandwidth. Data is trickled down to clients (both end user and to the Shell servers) effectively as background processes, allowing normal operation to take place concurrently. All data packets are compressed before being packaged within JMS. Such decoupling also provides resilience as component failure; the failure of a single Shell server should not effect the operation of others.

In summary, the NFFS architecture provides a controlled runtime framework for the associated modules. It ensures that Module processes are managed and that data and module configurations are held within a controlled environment.

B1.3 Physical mapping of roles in the NFFS

Figure B1.3 provides an overview of the main components within NFFS.

The MasterController (MC) components are the heart of the server. The MC is responsible for centralized data storage, data synchronisation. The MC contains a TaskManager which maintains a task list and schedules and dispatches tasks to shell servers via JMS.

The Operator Client (OC) components hold the GUI components for presentation. Data is presented from a local data store which, by messaging with the MC, is continuously synchronised with the central data store.

The FEWS Shell Servers (Shell servers) executes tasks and runs the numerical models via adaptors. The tasks are run from a local data store which is synchronised, by messaging with the MC, with the central data store. Results data and logging are handed back to the central system via JMS.

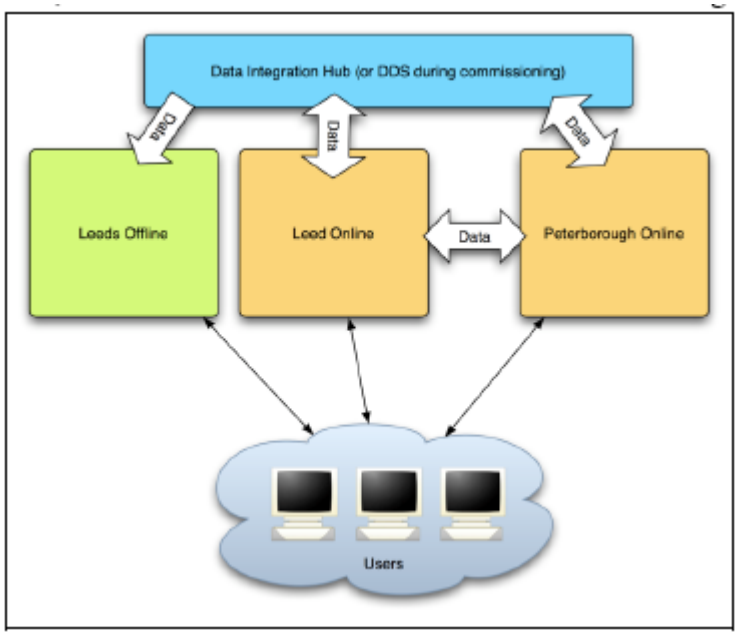


Figure B1.3 Main components within the NFFS architecture

B1.4 Installations

The NFFS consists of three complete installations of the architecture (see Figure B1.4):

Online system (Leeds)

- Used for operational forecasting
- The “production system”

Online system (Peterborough)

- Same as Leeds
- Both sites are synchronised with each other

Offline system (Leeds)

- Used for testing and calibration, acceptance of new models etc.

Live data feeds from Telemetry and the Met Office are provided by the Data Integration Hub, which also acts as a distribution mechanism for NFFS generated forecast time series.

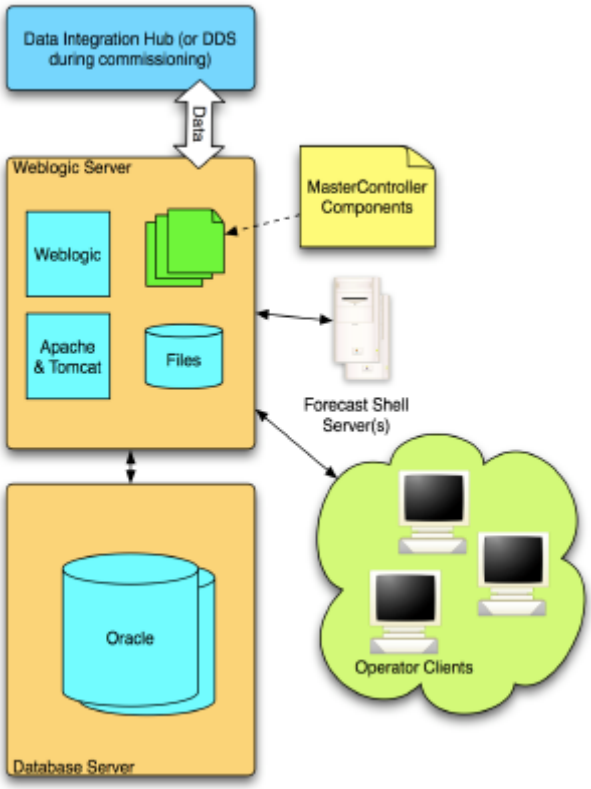


Figure B1.4 High level view of the implementation of NFFS within EA

B2 NFFS TRITON Module – Case Study

B2.1 Overview

This section aims to serve as a guide to producing a module which can be integrated into the National Flood Forecasting System (NFFS), currently being implemented for the Environment Agency.

It uses an existing module, TRITON, as a case study. This module is currently part of the live NFFS system and was developed using the guidelines produced jointly by the EA and Delft Hydraulics. Because there are some key elements to module functionality which are not used by TRITON, the document also refers to a second module (PRTF) when discussing these extra features.

One of the main aims of the NFFS project was to define standards to enable 3rd. party developers to produce modules which could seamlessly be integrated into the NFFS. The five key requirements for a module are: -

- Ability to receive data from NFFS in a published standard format.
- (Optionally) convert this data into a format which the module can use.
- Manipulate the (converted) data as part of the module process, producing a set of results (typically this would be a forecast).
- Convert the results back into the published format for NFFS.

- Return module diagnostic information back to NFFS.

A module can be considered to be a "black-box" which is invoked by NFFS as required. NFFS is the custodian of all the input data a module normally requires and it is configured to deliver this data to a location which can be accessed by the module. Once this is done, NFFS activates the module and waits for it to complete, at which point NFFS fetches the output data from the module, which was placed at another common location.

Modules may have local configuration information and other bespoke data which forms part of the module data set. A module is configured to perform a normal (default) set of tasks, using this module configuration. There is no direct human interaction with a module but there is a mechanism whereby users can override the default behaviour of the module, using NFFS to specify the required overrides. NFFS conveys any changes from the default behaviour via a parameter file encoded in a published format.

Finally, for modules which use the outputs from the previous run to initialise the next run, NFFS is able to store this information and deliver the required files to provide a starting State for the module.

B2.2 Module Adapters

The TRITON module performs all 5 tasks listed above, in the order specified.

However, another approach which can be taken is the Module Adapter method. An adapter is effectively some specialist software which converts inputs from NFFS into a format which the module can use and then, once the module has been completed, converts the bespoke format module outputs back into a format which is supported by NFFS.

An Adapter is really just another process which must be invoked before the module is run and then after the module has completed. In the case of TRITON, the data conversion is performed by the same executable as the module itself - the Adapter forms part of the module.

Either method is supported. It may be that it is not convenient or possible to modify the module code so a separate task is required to convert the NFFS data into the format the module expects.

The flow-chart below shows the module-adapter approach.

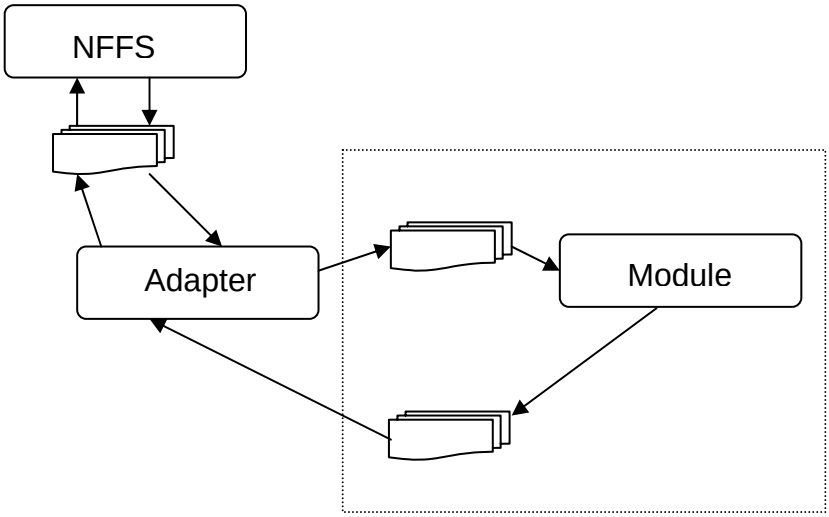


Figure B2.1 - Adapter - Module - Adapter Method

By following the arrows, it can be seen how NFFS exports data in one format, the Adapter converts this to something that the module can deal with, the module works on this data and exports results in its own format which are finally reformatted by the Adapter into NFFS standard.

Figure B2.2 shows the TRITON (no adapter) method where the module is also the adapter.

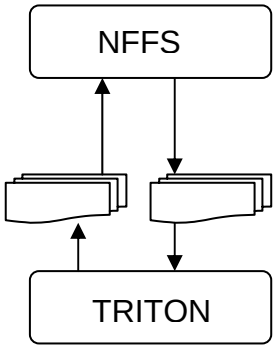


Figure B2.2 - Module only Method

B2.3 Data Exchange

Figures B2.1 and B2.2 show the two possible methods whereby data can be exchanged between NFFS and a module. It is normal for NFFS to place the required data into a disk location which is within the module scope. Likewise, NFFS will expect to find the module results (in NFFS format) and the diagnostic file in a different location, once again within the scope of the module. This can be thought of as an "inBox" and "OutBox".

The module (or adapter) should be configured to read from the InBox and write to the OutBox and these should ideally be set up as relative paths to where the module is installed. Absolute paths should not be used anywhere in

the module - this allows the module to be located on any server, without having to rely on paths being specifically configured.

For example, the TRITON hierarchy is as follows: -

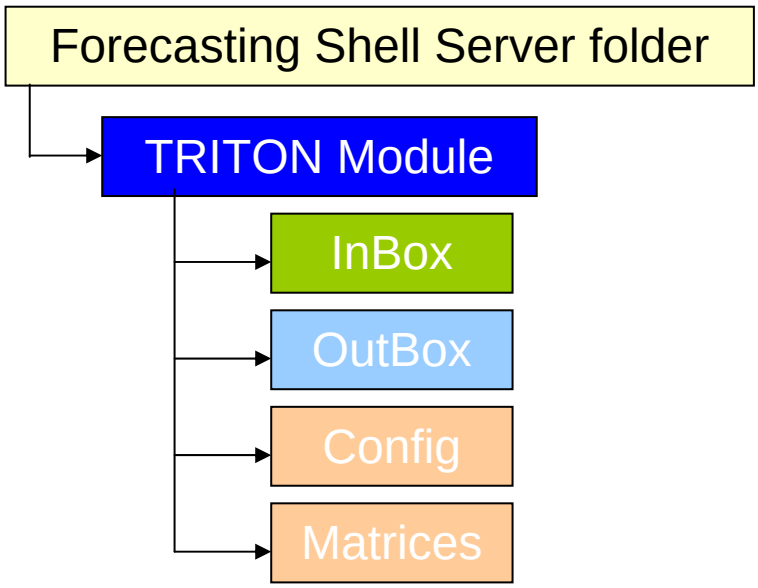


Figure B2.3 - TRITON folder structure

Before it calls the TRITON module, NFFS performs the following tasks: -

- Empties the InBox and OutBox folders
- Places all the NFFS data required by the module into the InBox
- Places any Parameter data (to override the default behaviour) into the InBox

NFFS then invokes the TRITON module by running the TRITON executable which is located in the "TRITON Module" folder.

NFFS waits for the TRITON module to finish. It is very important that a module completes in a controlled fashion and does not "hang". NFFS will have a fail-safe time-out which it will use to terminate the process if it has not stopped after a pre-determined (configurable) time.

Once the module has completed, NFFS then does the following: -

- Copies all the files from the OutBox to NFFS folders for processing
- Uses the information in the diagnostic file to provide feedback on-screen to the user and store in the NFFS logs.
- Makes the TRITON forecasts available on NFFS.

The TRITON Module uses the OutBox folder to communicate results back to the calling module (NFFS). NFFS takes responsibility for clearing out this folder to ensure that no data remains from the previous module run, but for completeness, the TRITON Module also ensures that there is nothing in the

OutBox before it starts. On completion of a run of the TRITON module, in this folder there will normally be: -

TRITON_diag.XML
TritonForecasts.XML
TritonForecasts.BIN

In the rare event that the module fails to complete, has insufficient data to initialise, the configuration is incorrect or expected files are missing, then only the TRITON_diag.XML file will be present. It will contain a set of log messages generated by the module, each with a certain severity (from diagnostic to critical error) which can then be analysed by NFFS and presented to the end user.

B2.4 NFFS data formats - Published Interface

As detailed in the above sections, it is important to note that NFFS will use a standard format to deliver data to any module and also expects the module to standardise on the outputs placed in the OutBox.

The NFFS Published Interface (PI) can be obtained from Delft Hydraulics or the Environment Agency in PDF format. It can be downloaded from the DH Web Site.

XML is used as the approved data exchange method. There are 14 XML schemas which comprise the Published Interface. Whilst at first glance, the PI can seem somewhat daunting, in reality, most module developers will only need to use a small sub-set of the 14 schemas.

The most commonly used is the "pi_timeseries" schema. TRITON also uses "pi_parameters" and "pi_diag". All three of these are covered in this document, although it is not intended for the reader to use this case study to learn how to use the PI.

TRITON uses the schemas as follows: -

- pi_timeseries - raw data required by the module, forecasts returned by the module
- pi_parameters - overrides set up by the user, via NFFS, to instruct the module to perform something other than the default behaviour.
- pi_diag - a type of report which is returned to NFFS

There can be a large amount of data provided by NFFS to TRITON (or any module). Whilst XML can be ideal for the exchange of data in a controlled and safe way, it does have the overhead that the files can be quite large, considering the relatively small amount of data you may extract from them. In the same way, the outputs from a module may also be considerable. For example, one configuration of TRITON results in forecasts being produced for more than 200 locations and each forecast in turn comprises more than 10

parameters. The outcome is that a file which is almost 50 Mb can be produced.

Whilst this may not necessarily be a problem if the module is only run e.g. every 12 hours, there may be a penalty to pay in response time if the NFFS user has asked for a forecast and urgently needs to see the results (perhaps as the result of a "What If").

Fortunately, the NFFS published interface allows large datasets to be transferred in binary format, with the key information only being listed in the pi_timeseries XML. The use of binary is covered later in this document.

B2.5 Data exchange - not Published Interface

Some modules use state updating (i.e. the output from one run is normally used to initialise the next run). It cannot be assumed that a module will always be invoked at regular intervals, however desirable that may be. They will be times when maintenance is being carried out or a server fails and one or more state runs are not carried out.

As a result, it is not recommended that a module uses any locally stored dynamic data to initialise.

The preferred method is that at the end of a run, the module returns the module state, with relevant time-stamps (in bespoke module format if this is necessary) to NFFS, via the OutBox. NFFS does not actually process this data (and besides, it may be in bespoke format). Instead, it can be configured to store the data in an archive for subsequent retrieval as the initial state for the next run. It can also be used to initialise hindcast runs, where the state to be used for a one-off run may be from several days ago.

Whilst TRITON does not use this method, a small example may be useful to explain how NFFS can provide module state data.

Consider a module which runs every 12 hours, at 01:00 and 13:00.

Assume that the first run takes place at 01:00 on October 1, providing a forecast from 00:00 October 1 to 23:00 on October 2.

As well as returning the forecast to NFFS, the Module would return any data it would need to initialise the 13:00 run on October 1. This could in fact be the whole 01:00 forecast or a subset of the data.

NFFS makes the "proper" forecast available to the end users but also stores away the 00:00 October 1 state.

At 13:00 on October 1, NFFS retrieves the 00:00 October 1 state and places it in the InBox for the module, along with any (PI) data required by the module. Once again, the module generates a forecast and a state for 12:00 October 1.

In effect, NFFS ends up with a series of states, as well as the forecasts.

Perhaps on October 5th, you decide to rerun the October 1, 12:00 forecast. NFFS can retrieve the October 1, 00:00 state and provide this to the module, unless you decide to reset the module, in which case NFFS would not provide any state information.

B2.6 Module Dataset

As well as the dynamic data provided in the InBox by NFFS, a module will require other static data to complete the process. This data is normally referred to as the Module Dataset. There are no limitations on how much of this data there can be, which (Module) folders you put it in, the format you use etc. The important point, however, is that the data is static and the intention is that it is not frequently revised as this goes against the edict that modules (along with their datasets) should be thoroughly tested off-line before they are installed in the "Live" NFFS folders.

The same applies to any Dynamic Link Libraries (DLLs) which the module executable may use - you cannot expect to update these on an ad hoc basis as there are strict controls over how and why new releases of modules can be issued.

We can use TRITON as an example to demonstrate a module dataset. Consider the folder structure from Figure B2.4 and ignore the InBox and OutBox as they do not form part of the module dataset.

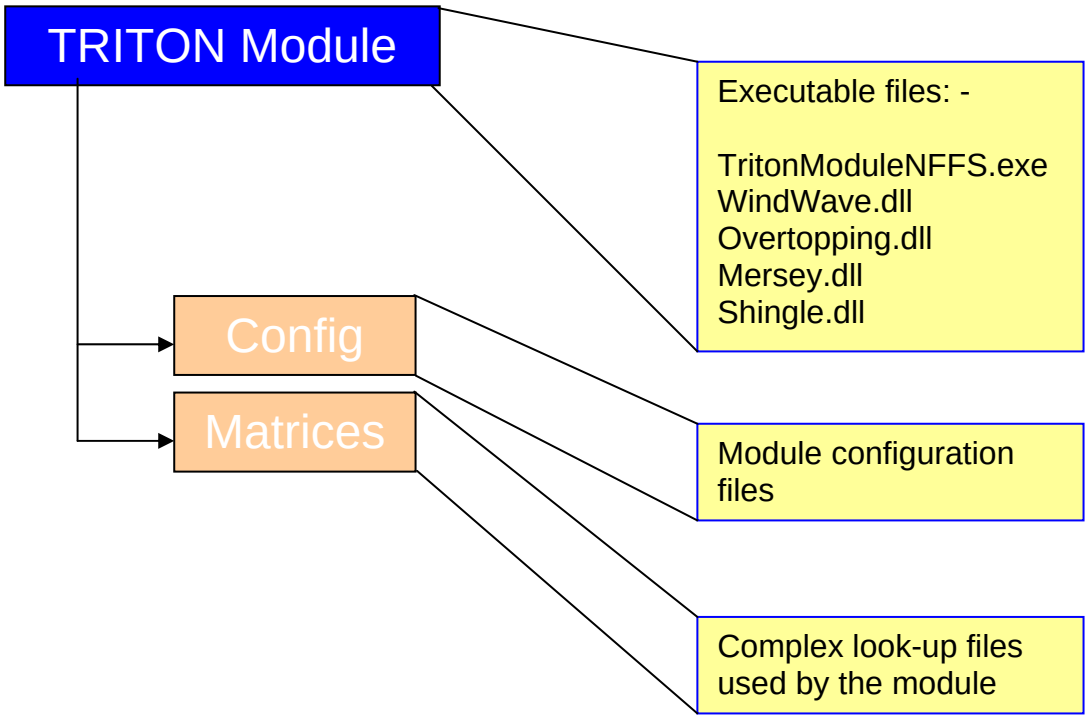


Figure B2.4 - TRITON Module Dataset

The **Executable** files consist of the main EXE file which is called by NFFS when the module is to be run, combined with some DLLs which are required by the module.

Note that there are a number of different instances of the TRITON Module (for different regions of the EA), each of which has some common executable code (TritonModuleNFFS.exe is the same for all instances) and also some regional specific DLL code (e.g. Mersey for NW Region and Shingle for Southern Region).

Likewise, the **module configuration** files are Regional specific (or really, installation specific). It may be that the module serves only one specific purpose, in which case there may only be one set of configuration files. In the case of TRITON, the configuration files vary from one installation to the next.

The Config folder is mandatory for all instances of the TRITON Module. It contains a variable number of files, depending on the region. Some files are present in all cases, others are common to more than one (but not all) installation / region and finally, there are some files which are unique to a specific region. TRITON has been designed to use .INI file formats to store configuration data to make it easy for the files to be maintained. Note that all of the module configuration data files are read only - they contain information which drives the module in default mode. Typically this will be about the forecast locations and data specific to those as well as a list of input data which is expected from NFFS so a sanity check can be performed.

A more detailed description of the TRITON Module Configuration can be found in the document "*Triton Module Configuration*".

For some regions only, the TRITON Module uses Matrices as complex look-up tables to convert Offshore Wind and Wave conditions to Nearshore equivalents and Nearshore values into Overtopping. In those installations, the matrices folder must exist and contain the relevant matrix files.
In summary, the Module Dataset comprises all files (executable and data) which the module needs to manipulate the input data provided by NFFS, producing results which are returned to that system.

B2.7 Module input data

NFFS will be configured to place all of the input data required by the module in a disk location which is easily accessible by the module. This task is completed before the module is activated. It is the responsibility of the module to ensure that all the required data is provided and any missing or erroneous data issues should be reported back to the calling process (NFFS), using the Diagnostic XML output file.

In some circumstances, depending on what is wrong with the input data, it may be possible for the module to complete a full or partial run. Even in those circumstances, it is important that the module reports back to NFFS that there were problems with the data provided. It is very likely that subsequent runs

will have similar problems and it may be able to be fixed by a change to the NFFS configuration.

As discussed in Sections B2.4 and B2.5, the inputs from NFFS to any module will conform to the Publish Interface standards and in some circumstances they may also involve the delivery of state information (the output from a previous run of the module), which is in a format which is bespoke to the module. It should be emphasised that the only files which can be transferred between NFFS and the module which do not meet the PI standards are those which are not actually read or written by NFFS - they are simply stored in an archive for future delivery to the module as initialisation data for the next run.

The following sections should also be read in conjunction with the NFFS Published Interface standards document. These sections only detail the use of the files within TRITON but should serve as a useful introduction to data exchange.

B2.7.1 NFFS Time-series data

This type of XML file forms the basis of most of the data transfer into and out of modules. The XML Schema definition file is called **pi_timeseries.xsd.**

They are used to transfer time-series data into the module. Note that NFFS can be configured to place all of the time-series data into a single input file or, more likely, it will be split into several files.

In the case of TRITON, two XML files are provided.

- TIDE_DATA.XML
- WIND_DATA.XML

The TRITON module needs Astronomic Tide data, Surge forecasts, Wind and Wave forecasts from a number of locations. The Astronomic and Surge data is provided in TIDE_DATA.XML and the Wind and Wave forecasts are delivered to the module in WIND_DATA.XML

It is important to note that the module should not "Hard-code" any input filenames as this prohibits flexibility. Likewise, it should not expect certain time-series in particular files.

Instead, it should be programmed to read all of the XML files found in the InBox and handle what it finds. This is one of the benefits of using XML.

As an example, the TRITON module would still work if the NFFS configuration was changed to either: -

- Provide all the data in one file
- Split the Tide and Wind data into multiple files
- Put some of the Astro Time series data into WIND_DATA.XML

Sample extracts from the TIDE_DATA.XML file are given.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <TimeSeries xmlns="http://www.wldelft.nl/fews/PI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance" xsi:schemaLocation="http://www.wldelft.nl/fews/PI
      http://fews.wldelft.nl/schemas/version1.0/pi-schemas/pi_timeseries.xsd" version="1.2">
    <timeZone>0.0</timeZone>
  - <series>
    - <header>
        <type>instantaneous</type>
        <locationId>ABDN</locationId>
        <parameterId>Tide.astronomic</parameterId>
        <timeStep unit="second" multiplier="900" />
        <startDate date="2006-08-29" time="00:00:00" />
        <endDate date="2006-08-30" time="12:00:00" />
        <missVal>-30000.0</missVal>
        <longName />
        <stationName>Aberdeen</stationName>
        <units>m</units>
        <sourceOrganisation />
        <sourceSystem />
        <fileDescription />
        <region />
      </header>
      <event date="2006-08-29" time="00:00:00" value="-0.106" flag="1" />
      <event date="2006-08-29" time="00:15:00" value="0.072" flag="1" />
      <event date="2006-08-29" time="00:30:00" value="0.252" flag="1" />
      <event date="2006-08-29" time="00:45:00" value="0.431" flag="1" />

                  |      |                    |          |

      <event date="2006-08-30" time="11:30:00" value="-0.871" flag="1" />
      <event date="2006-08-30" time="11:45:00" value="-0.792" flag="1" />
      <event date="2006-08-30" time="12:00:00" value="-0.697" flag="1" />
    </series>

</TimeSeries>
```

Key points to note: -

- The Schema definition file is pi_timeseries.xsd
- The file comprises one or more <series>
- Within each <series>, there is a <header>, followed by event data.
- The <header> describes the event data that follows and in the above example, it can be seen that the data is Astronomic Tide data in metres from ABDN (Aberdeen), it is equidistant with each data point being 900 seconds (15 minutes) apart. The data ranges from 29 August 2006 00:00 to 30 August 2006 at 12:00. Missing data will be given as -30000.
- Each Astronomic tide value is date and time stamped

There is enough information in this block for a module to extract and validate the Astronomic Time Series for Aberdeen. There would be multiple <series> in the file, normally. It is completely the responsibility of the module to ensure the integrity of the data.

Warning - do not make the mistake of treating XML files as simple ASCII. Use a proper XML Parser to allow flexibility and structural change. Never assume some information will be e.g. on line 6, column 15 - this is asking for problems.

The above example shows equidistant data over a 36-hour period. TRITON also uses non-equidistant time-series data from locations where a full 15-minute Astronomic Tide series is not available (only High Tides or High and Low Tides). See the following example: -

```
<?xml version="1.0" encoding="UTF-8" ?>
- <TimeSeries xmlns="http://www.wldelft.nl/fews/PI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:schemaLocation="http://nffs.wldelft.nl/schemas/PI http://nffs.wldelft.nl/schemas/pi-
    schemas/pi_timeseries.xsd" version="1.2">
  - <series>
    - <header>
        <type>instantaneous</type>
        <locationId>BIDE</locationId>
        <parameterId>Tide.astronomic</parameterId>
        <timeStep unit="nonequidistant" />
        <startDate date="2005-04-12" time="20:00:00" />
        <endDate date="2005-04-13" time="20:30:00" />
        <missVal>-30000</missVal>
        <units>m</units>
      </header>
      <event date="2005-04-12" time="20:00:00" value="5.370" flag="2" />
      <event date="2005-04-13" time="08:15:00" value="5.080" flag="2" />
      <event date="2005-04-13" time="20:30:00" value="4.880" flag="2" />
    </series>
</TimeSeries>
```

In this case, the timeStep unit is set to "**nonequidistant**", the start and end date reflects the 3 high tide values which are available.

B2.7.2 NFFS Binary data

In some cases, the input data to the module may be considerable and very large XML files are necessary to convey all the data. It can be seen that, whilst XML is an ideal standard for data exchange, there are large overheads in data size and also the time taken to parse the files should not be under-estimated.

Each value in a simple 36 hour, 15 minute resolution time-series is date and time-stamped for completeness and to meet the schema standard. This in itself adds a large overhead, especially as all you would normally want would be the start date/time, end date/time and the 144 values, probably around 1200 bytes of data in total (144 * 8 bytes for a real number) + 2 dates / times.

NFFS provides a way to combine XML and binary which can be used both with exports from and imports to the NFFS.

Consider this simple example (PRTF): -

```
- <series>
  - <header>
      <type>accumulative</type>
      <locationId>208</locationId>
      <parameterId>Rainfall</parameterId>
      <timeStep unit="second" multiplier="900" />
      <startDate date="2006-10-06" time="11:00:00" />
      <endDate date="2006-10-14" time="11:00:00" />
      <missVal>-30000.0</missVal>
      <longName />
      <stationName>Vallis</stationName>
      <units>mm</units>
      <sourceOrganisation />
```

```
        <sourceSystem />
        <fileDescription />
        <region />
      </header>
    </series>
  - <series>
```

Everything in the series header looks the same as we would normally expect, equidistant time-series at 15-minute resolution etc.

The main difference, however, is that there is no event data after the header. This data is provided in a separate binary file. These main points are relevant if binary is being used as an input method: -

- In the InBox, for every XML file, there will be a matching '.BIN' file with the same prefix.
- Binary can only be used for equidistant data.
- Parsing the XML file from top to bottom, each time a series header is read, you must read the appropriate number of binary values (stored as floating-point numbers) from the .BIN file. The number of values to read has to be determined from the startDate, endDate and timeStep in the header. In the example given, this equates to 7 days = 7 * 96 values.
- Note that there is no header information in the .BIN file. It starts straight away with the first value from the first time-series.

B2.7.3 NFFS Parameter data

As stated in Section B2.6, the Module dataset, and in particular the Configuration, should be set up so that once the module is invoked by NFFS, the input data read and validated, the module knows exactly what to do with the data in order to provide the results required by NFFS. This is how TRITON works.

In effect, TRITON does not care too much about the input data values (they may have been adjusted in some way in NFFS, perhaps a user has added a constant to each surge value). Provided all the required time-series are received, it uses that information to produce a set of forecasts which it returns to NFFS, via the OutBox (See Section B2.8).

A module is only under control of NFFS (there is no User Interface to the module) so if you want to temporarily change the default behaviour of the module, then this needs to be done via NFFS. Where a module can be "controlled" in some way by NFFS, this is described as running a "What If".

For each remotely configurable aspect of a module which is supported, NFFS can be configured to provide a user-interface to simply allow an override to be specified. If any such changes are made, they are only temporary and for the current module run.

These one-off requests to the module are conveyed in parameter files, conforming the pi_parameter standard defined in the NFFS Published

Interface. If your module allows remote configuration of one or more parameters, then the module should be able to process these parameter files.

As with Input data, the parameter files will be placed by NFFS in the InBox, the module must be able to differentiate between parameter files and time series data and no assumptions should be made about the names of the parameter file or files. Total flexibility should be programmed in, including the possibility that there will not be any parameter file (in which case the default behaviour is performed).

Consider the following example from the PRTF module: -

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Parameters xsi:schemaLocation="http://www.wldelft.nl/fews/PI http://nffs.wldelft.nl/schemas/pi-
      schemas/pi_parameters.xsd" version="1.2" xmlns="http://www.wldelft.nl/fews/PI"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   - <param id="311">
        <name>modelname</name>
      - <data>
           <stringData>Wet</stringData>
        </data>
     </param>
</Parameters>
```

Note firstly the name of the schema - **pi_parameters.xsd**

In this case, the <param id="**311**"> node is specifying an override of the default behaviour for one of the forecasting sites (ID = 311). Site 311 (Bishops Hull) normally uses Catchment Wetness index (CWI) at the point where the river level is lowest (base flow) to decide which rainfall runoff model to run (e.g. Very Dry, Dry, Wet, Saturated, High Intensity).

It may be that the user wants to force the module to run the Wet model, in spite of what the CWI may indicate is preferred. This selection could be made via the NFFS Interface and a parameter file such as the above would be delivered to the PRTF module along with the input data.

It is the responsibility of the module to detect and process the contents of these parameter files and to use the information contained therein to drive the current module run. In this example, the user would expect to see that the PRTF module had run the "Wet" model for the Bishops Hull forecasting site (ID = 311).

B2.8 Module output data

This Section should be read in conjunction with Sections B2.7.1 and B2.7.2.

Once NFFS has delivered all the module input data and any parameter files, called the module, it will then wait until the module has completed before looking in the OutBox for the module output files which it will then move to the NFFS system for processing and archiving as necessary.

These files should also conform to the same standard as the pi_timeseries XML files + Binary files which are imported by the module.

Normally a module would return a number of time-series for each forecasting location. These can be arranged in any order and may be placed in a single XML file or in multiple files. As with the Input files, there are no restrictions on the file names you use as NFFS will process all files added to the OutBox.

If there is a lot of data, it is recommended that binary is used. An example section (single series) from a TRITON output file is given (where binary is used).

```
<?xml version="1.0" encoding="UTF-8" ?>
- <TimeSeries xmlns="http://www.wldelft.nl/fews/PI" xmlns:fews="http://www.wldelft.nl/fews/PI"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.wldelft.nl/fews/PI http://fews.wldelft.nl/schemas/version1.0/pi-
        schemas/pi_timeseries.xsd" version="1.2">
    - <series>
        - <header>
            <type>instantaneous</type>
            <locationId>NE-Berw-Z1-St1</locationId>
            <parameterId>Calculated.Tide.Level</parameterId>
            <timeStep unit="second" multiplier="900" />
            <startDate date="2006-08-29" time="00:00:00" />
            <endDate date="2006-08-30" time="12:00:00" />
            <missVal>-30000</missVal>
            <stationName>Berwick Pier</stationName>
            <units>m</units>
          </header>
      </series>
  </TimeSeries>
```

In this sample, a Tide Level forecast (metres) at 15-minute (900 seconds) interval, over a 36-hour period for the location **NE-Berw-Z1-St1** is returned. Because binary is used, there is no associated data (see Section B2.7.2). Note that the standard is the same as that for the exports from NFFS.

In reality, this configuration of TRITON returns thousands of time-series (data from > 200 forecast locations, providing perhaps 10 parameters in each case). In this case, it is very wise to use binary, otherwise the XML file will be very large.

Note that the pi_timeseries schema does not support text in the "value" field. Whilst it is unlikely that text values will form part of a time-series, in fact within TRITON, this is necessary. TRITON makes a decision based on the input information as to which offshore wind/wave site to use. It needs to tell the end user which site was chosen. Because it cannot pass back the name (or site code), it was necessary to define a look-up / cross-reference table within NFFS which allows the module to return integers which represent the appropriate sites. These values are looked up in the tables by NFFS to extract the relevant site name as a text string.

There has to be a means of a module conveying status information back to the NFFS. Remember that the user will not normally "see" the module being run - it is run on a remote computer and may just be a console application which has no visible features at all.

There is a PI standard, pi_diag. Think of this as a simple log file, except using XML. Each line in the XML file contains a warning level and the text associated with it.

The important warning levels (extracted from the schema) are : -

3 = info (information, all is well, e.g., :"SOBEK: program ended")
2 = warn (warning information. e.g. "SOBEK: high number of iterations")
1 = error (critical problems. e.g. "SOBEK: no convergence")
0 = fatal (full module crash. e.g. "SOBEK: ooops, what now?")

Higher numbers than 3 can also be used - they will not be listed on the NFFS screens but may contain essential debug information.

The following are extracts from the PRTF diagnostic file. Note that some of the entries are generated as a result of the Parameter file discussed in Section B2.7.3.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Diag xmlns="http://www.wldelft.nl/fews/PI" xmlns:fews="http://www.wldelft.nl/fews/PI"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.wldelft.nl/fews/PI http://fews.wldelft.nl/schemas/version1.0/pi-
      schemas/pi_diag.xsd" version="1.2">
    <line level="3" description="Starting PRTF Module at 13-Oct-06 13:41:49.515" />
    <line level="3" description="Site Bishops Hull - default model changed to Wet" />
    <line level="3" description="XML InBox processed" />
    <line level="3" description="Processing Bishops Hull" />
    <line level="3" description="PRTF SMD valid..." />
    <line level="3" description="Calculating SMD using continuous update method using AE MORECS" />
    <line level="2" description="No SMI value to start sequence - using 0.5" />
    <line level="3" description="Generating Outputs" />
    <line level="3" description="Exiting PRTF Module at 13-Oct-06 13:42:06.031" />
</Diag>
```

Note the level 3 entry which confirms that the module recognised the request via a parameter file to override the default model and use the Wet Model.

There is only one Level 2 (Warning) which tells the user that there was not any Soil Moisture Index data provided to the module so it defaulted to a value of 0.5.

A module should always generate a diagnostic file, even if it just has one entry in it to say that it completed successfully.

B2.9 Other Module requirements

Unless a module is being adapted from an existing stand-alone application and it is difficult to do so, an attempt should be made to create the module as a console application. In other words it does not use any Windows, graphics, user interface etc. Apart from the fact that the module runs remotely so there is no point in there being any graphics or visible elements whatsoever, a console application will run much more quickly.

It is vital that every attempt is made to ensure that a module completes in a tidy controlled fashion. NFFS starts a module then monitors the operating

system to see when it completes, at which point it fetches the outputs and makes the results available within NFFS. If a module "gets stuck", then NFFS will think it is still busy and keep waiting for it to complete. There is a setting which can be set up in NFFS which is a time-out, after which NFFS assumes the module is not going to complete. As a fail-safe, this has to be considerably higher than the normal time taken to complete, to avoid NFFS stopping the process when it is running normally.

If the module hangs, then the users will have to wait a long time to have this information and then they will discover that the forecasts were not generated anyway.

Good coding practice should be adhered to throughout the module - basic checks such as trapping divide by zeros, file reading / writing errors etc. should all be included. The module should never assume that all the files it needs are always present, it should check each time.

Ideally, the software should always come to a controlled stop and tidy up itself (at the very least write the diagnostic file), even if the fundamental part of the processing fails. This can be achieved by good coding, even it means putting an exception handler around the main entry point to catch any problems.

References

Agency documents

Enterprise Architecture Approach for the Environment Agency (PowerPoint presentation – 1st February 2006) – Ash Dattani

Enterprise Architecture: Technical Reference Model (Version 2.0 DRAFT3 - 6/12/05) – Peter Wintle, Enterprise Architecture

Environment Agency Testing Method (Version 1.0 Final – 15/06/2005)

Technical Delivery Process Document (Version 1.3 - 23/08/2005)

Other

http://www.govtalk.gov.uk/schemasstandards/schemasstandards.asp (including e-Government Interoperability Framework (e-GIF), Version 6.0, 30 April 2004)

OpenMI standards – http://sourceforge.net/openmi

Oracle 10g Express Edition
http://www.oracle.com/technology/products/database/xe/index.html

JUnit Java unit testing - http://sourceforge.net/projects/junit/
NUnit .NET languages unit testing – http://www.nunit.org/

XML standards - http://www.w3.org/TR/2004/REC-xml-20040204/

Glossary of Selected IT Terms

ASP – Active Server Pages
HP-UX – Hewlett Packard Unix
IIS – Internet Information Server
J2EE - Java 2 Platform, Enterprise Edition
JSP – Java Server Pages
LAN – Local Area Network
TCP/IP - Transmission Control Protocol/Internet Protocol)
XML – Extensible Markup Language
WAN – Wide Area Network

PB 12794 A

Ergon House
Horseferry Road
London SW1P 2AL

www.defra.gov.uk