Deliverable Title	**D6.1.2 Specifications for Experimental Metadata (Version 2)**
Deliverable Lead:	IT Innovation
Related Work package:	WP6
Editor(s):	J Brian Pickering (ITI), Steve Taylor (ITI)
Dissemination level:	Public
Due submission date:	31/08/2011
Actual submission:	22/09/2011
Project Number	258142
Instrument:	IP
Start date of Project:	01/06/2010
Duration:	30 months
Project coordinator:	THALES

Abstract	This document represents an update on the initial TEFIS Data Services approach outlined previously [1]. With the benefit of direct experience with the initial TEFIS use cases, we outline the use-case specific requirements and how they are being addressed during the implementation of the TEFIS Data Services. Metadata models, for curation and search, are outlined as well as the physical storage structure used to support the use cases and similar experiments throughout the experimental data lifecycle from experiment design to resourcing and finally to execution.

Project funded by the European Commission under the 7th European Framework Programme for RTD - ICT theme of the Cooperation Programme.

License

Versioning and Contribution History

Version	Date	Modification reason	Modified by
0.1	26/07/2011	ToC Proposal and initial outline	Steve Taylor (ITI)
0.2	23/08/2011	Initial Editing	Brian Pickering (ITI)
0.3	29/08/2011	Updating and restructuring	Brian Pickering (ITI)
0.4	12/09/2011	Version released to reviewers	Brian Pickering (ITI)
0.5	20/09/2011	Updated after SQS comments	Brian Pickering (ITI); Jonathan González (SQS)
0.6	21/09/2011	Updated after ITI internal review	Brian Pickering (ITI); Michael Boniface (ITI)
0.7	21/09/2001	Updated after ENG comments	Brian Pickering (ITI); Gabriele Giammatteo (ENG)
0.8	22/09/2001	Final version for submission	Brian Pickering (ITI)

Table of Contents

Table of Figures

List of Tables

Executive Summary

This document describes the 2nd version of specifications for experimental metadata used by the TEFIS Data Services. The document provides revisions to the first version of the *Specifications of Experimental Metadata* [1] based on the experience gathered as part of the initial TEFIS Platform implementation in support of the first of the TEFIS use cases, eTravel.

The TEFIS Data Services are a central, and one of the most significant components in the TEFIS platform. They are responsible for maintaining data for the TEFIS experimenters, data associated with and generated by the experiments they run, and data from other experimenters that they may wish to consult when planning their own work. The TEFIS Data Services are also responsible for providing the data handling for all of the TEFIS internal components, for them to be able to store the data they need to be able to process user experiments, and for them to be able to act on the requests of the TEFIS experimenters. Finally, the TEFIS Data Services are responsible to the TEFIS testbed providers, in allowing them access to the data they need to satisfy requests to execute experiments, as well as providing them a location to store output data from the tests they run. In short, the TEFIS Data Services are significant for all of the TEFIS stakeholders in support of the smooth operation of the platform and of the experimenters who use it.

Initial experience in the first year of the project and in respect of the specification and execution of the first of the TEFIS use cases has led to the revision of the original data model. The revisions to the data model include the following:

- The initial test cases, which address eCommerce, eHealth and multi-media services, have been examined in greater detail and with specific reference to their modelling requirements from the perspective both of experimenter as well as testbed resource provider, which has led to the separation of experimenter-focused tasks (*what* is being tested) and testbed-centric execution instances (*how* it is being run);
- Existing experimental and scientific metadata standards, including the Dublin Core and the Core Scientific Meta-data Model, have been investigated and used as the basis for a comprehensive metadata schema associated with running experiments across heterogeneous testbeds;
- The domain model has been separated into three sub-domains associated with the main phases of the experiment lifecycle:
    i. *Design*: covering the actual definition of the overall experiment and any individual test runs or experiment instances;
    ii. *Resourcing*: including the identification of appropriate resource to be able to support the execution of a given experiment instance; and
    iii. *Execution*: to capture all aspects of the running of experiments.

- A revised physical data model has been developed to enable the more effective support of all the TEFIS stakeholders, including a physical folder structure that separates TEFIS-internal and experimenter data elements for easier management of the experimental lifecycle.

This deliverable therefore marks a revision to the first version of the *Specifications of Experimental Metadata* from the first period of the project ([1]) in light of experience from that period, as well as how the TEFIS Data Services are being implemented for the remainder of the project with a view to a more effective support of the main TEFIS stakeholders operating across heterogeneous test facilities whilst ensuring data integrity and security.

1. Acronyms and Abbreviations

CSMD	*Core Scientific Meta-Data Model*
DCMI	*Dublin Core Metadata Initiative*
EI	*Execution Instance*
ETICS	*eInfrastructure for Testing, Integration and Configuration of Software*
EOL	*End of line*
FMF	*Full-Metadata Format*
GB	*Gigabyte*
ICT	*Information and Communication Technology*
ID	*Identity*
IMS	*IP Multimedia Subsystem*
iRODS	*Integrated Rule-Oriented Data System*
ISO	*International Organization for Standardization*
MDT	*Multi-disciplinary team*
NISO	*National Information Standards Organization*
PACA	*Provence-Alpes-Côte-d'Azur – the home of PACA Grid*
REST	*Representational state tranfer*
RPRS	*Research Platform Repository Service*
SLA	*Service Level Agreement*
SOA	*Service-oriented Architecture*
SQS	*Software Quality Systems*
STFC	*Science & Technology Facilities Council*
TCI	*TEFIS Connector Interface*
TIDS	*Testbed Infrastructure Data Service*

URL *Universal Resource Locator*

VCT *Virtual Customer Testbed*

WP *Work Package*

XML *Extensible Markup Language*

2. Introduction

During the first year initial versions of the TEFIS data services have been integrated and shown to work together with other components in support of the first TEFIS use case targeting eTravel. The TEFIS Data Service components comprise local structures, the Research Platform Repository Services (RPRS), which are designed to support the other TEFIS components during the preparation and execution of experiments; and remote structures, the Testbed Infrastructure Data Services (TIDS), to support both testbed providers and experimenters in their interactions with TEFIS (see, for example, Figure 1).

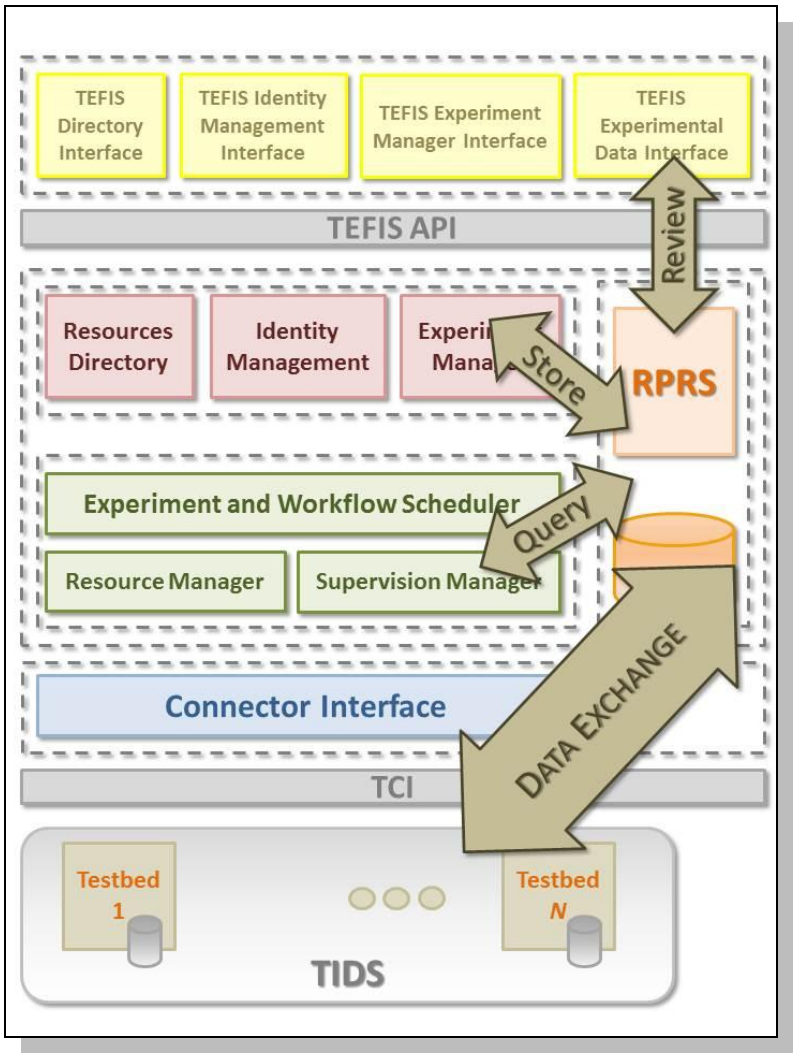

Figure 1: The TEFIS Data Services

The TEFIS Data Services are responsible for the management of all experimental data within and in association with the TEFIS platform. From the portal front-end itself, the TEFIS Data Services allow data to be searched and reviewed; for the back-end components providing the business logic for the portal, the TEFIS Data Services are responsible for storing and maintaining data about experimental design; for the core services, the TEFIS Data Services are the central, managed location where experiment-specific data are stored and from which the data required to initiate and run experiments can be retrieved; and finally, via the connectors, the TEFIS Data Services allow for the exchange of data between the RPRS and TIDS: to and from the testbeds.

The initial structures and metadata model proposed at the beginning of the TEFIS platform development [1] served as the basis for the first deployment. The components supporting the experimenter in the design, provisioning and deployment of experiments all made extensive use of the first TEFIS Data Services version, implemented as defined with a user- and experiment-centric folder structure and metadata on files and folders in that structure. The folder structure provided other TEFIS components with a central location in which to store and exchange data items needed to run experiments. For instance, the workflow specification that the experimenter created in the TEFIS Experiment Manager was stored in a specific location so that the Core Services could retrieve it and execute the task steps it contained. Similarly, as an experiment instance was executed, the testbed or testbeds running it would store output data from the experimental run in the specific location within the folder structure defined. So the folder structure provided a common, shared set of folders used to hold data needed to run an experiment or generated as a result of that experiment. As well as the folders and files themselves defined in this way, metadata items (attribute-value pairs) allowed experimenters to search through existing experiments to identify related work that they might be able to use as a starting point for or to validate their own work. This first implementation was described in [6] for the local data services (RPRS) and in [7] for the remote, testbed- or experimenter-side data services (TIDS).

As development has continued on the TEFIS platform, and the number of use cases to be supported is set to increase through the TEFIS Open Call, so the TEFIS Data Services have sought to review the design laid out in [1] and implemented as described in [6] and [7]. In the light of revised and extended requirements from experimenters, as well as from the other TEFIS platform components supporting those experimenters, and the testbed providers whose resource is made available through TEFIS, the original design has been reworked. In this document modifications to the original structures and metadata as a result of the experience of working with existing use cases are set out and discussed. Revisions to the original specifications are explained with reference to the three initial use cases (Section 4.1) and the experimental lifecycle that lies at the basis for TEFIS activities (Chapter 5). This has led to a revised data model (Chapter 6) more closely aligned with this experimental lifecycle and physical storage structure (Chapter 7) in support of the other TEFIS components and testbed providers as laid out below.

This deliverable therefore represents the current state of the TEFIS Data Services in light of revisions introduced as a result of

- experience with running the first test case (eTravel; see Section 4.1.1); and
- analysis of the other expected use cases (multimedia services in Section 4.1.2; and eHealth in Section 4.1.3) during the initial period of the project.

The specifications describe how TEFIS experimental data are to be managed for the remainder of the project including both the remaining use cases and those to be supported as a result of the TEFIS Open Call.

2.1. Document Scope

This is Deliverable D6.1.2, Version 2 of the Specifications for Experimental Metadata, which is an update to D6.1.1 [1] based on the experience of the first period. The document begins (Chapter 3) by revisiting the main TEFIS stakeholders and re-stating how the TEFIS Data Services seek to support them, as well as observations from the practical experience of implementation efforts thus far. Since one of the main objectives for TEFIS is to provide an appropriate access point to heterogeneous test facilities, we then look (Chapter 4) at the three initial test cases – eTravel (Section 4.1.1), a multimedia service (Section 4.1.2) and eHealth (Section 4.1.3) to define a topology of test types, and in the subsequent sections of that chapter, to outline the assumptions and consequences for specification of the TEFIS Data Services. In Chapter 5, we round off the initial discussions with a brief overview of the experimental lifecycle, first introduced in [1], and now as seen directly in relation to the initial TEFIS use cases.

The following chapters provide the implementation detail for revisions to the original specifications in [1]. First, Chapter 6 reviews the metadata models proposed for TEFIS in light of existing standards. The metadata structures used in TEFIS are primarily targeted to support users in their efforts to locate related work. The proposed metadata schema is reproduced for reference in Appendix I: *Metadata Schema* (Chapter 10), which bears some relationship to the emerging standards discussed at the beginning of this chapter.

This is followed in Chapter 7 with an introduction to the physical storage structures implemented in TEFIS and made available via the Research Platform Repository Service (RPRS, and as initially implemented in [6]) for Experimenters and TEFIS components to interact with, as well as via the Testbed Infrastructure Data Services (TIDS, as outlined in [7]) for Experimenters and Testbed providers. The TEFIS Data Services have been implemented using iRODS[1]; the justification for this technology selection and in particular in respect of the requirements first outlined in [1] is presented in Appendix II: *Technology Choice* (Chapter 11).

[1] www.irods.org

3. Background: Moving Forward with the TEFIS Data Services

The TEFIS platform provides a single access point for the design, management and support of experiments using heterogeneous test resources. There are three main stakeholders with interests in and requirements from the platform; these are summarised in Figure 2. As they interact with the platform, the different stakeholders have different types of requirement.

Figure 2: the TEFIS Stakeholders

TEFIS Operator	Those responsible for the continued operation of the TEFIS platform, including the components within the TEFIS architecture (see Figure 1).	Needs to ***RUN*** and maintain the TEFIS platform. From a data perspective, this includes having appropriate access to data associated primarily with the experiments executed on testbeds accessible through the TEFIS platform.
Testbed Providers	Those who provide testbed resources via the TEFIS platform, connected with it via the TEFIS Connector Interface (TCI).	Need to ***SUPPORT*** the experiments being run via the TEFIS platform. For the TEFIS Data Services, this means that all types of data – user information, configuration data, application data, performance data and research results (see [1], Table 2) – must be accessible to the testbed providers or will be generated or updated by them. The TEFIS Data Services must allow appropriate access to data items as well as data structures for the running of experiments, therefore.

Experimenters	The community of those who use the TEFIS platform in search of related work as well as to gain access to appropriate testbed resources.	Need to ***USE*** the TEFIS platform in support of their experimental research.

4. Use Case Analysis & Requirements

As an access point for running experiments across heterogeneous test facilities, TEFIS supports two main external stakeholders: those running experiments (*Experimenters*) and those supporting them (the *Testbed Providers*). Figure 3 lists the main use cases associated with the external stakeholders.

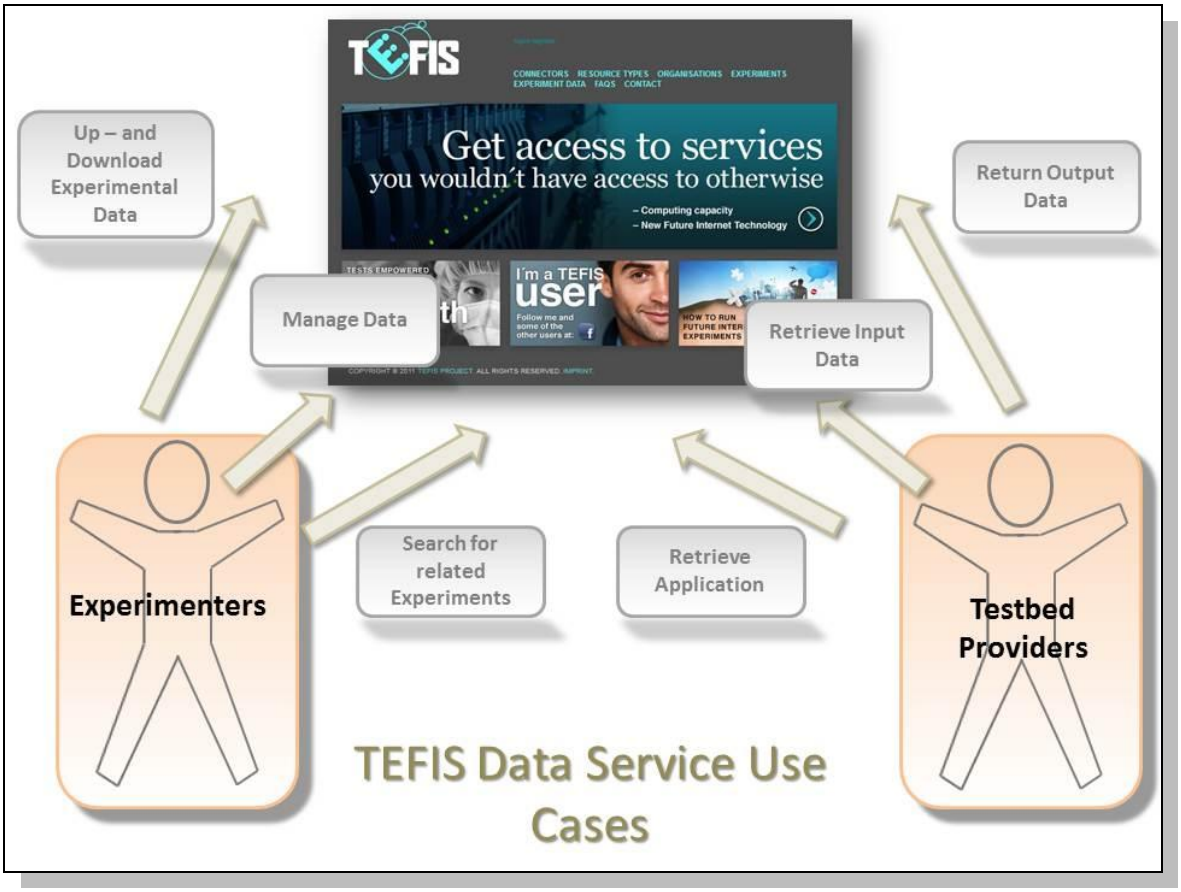

Figure 3: the main external stakeholders and use cases associated with the TEFIS data services

These include:

- For the *Experimenter*
    - *Up- and download experimental data*: the experimenter needs to be able to provide data appropriate to the platform to describe and be able to run the experiment instance; and once the experiment completes, they need to be able to download the results (including any monitoring output);

- *Manage data:* the experimenter wants to access his/her own data, transfer it elsewhere, or even delete it; they may also wish to allow access to their data or any specific data objects to other experimenters;
- *Search for experiments:* to get started with a new or modified experiment, the user needs to be able to find their own experiments, or may wish to see if they can find experiments from others[2].

- For the *Testbed Provider*
  - *Retrieve application*[3]*:* for a test to be run, application code or executables need to be retrieved for processing in preparation for or as the experiment;
  - *Retrieve input data:* to run a given experiment, as well as an application code or modules, there may be application or configuration settings or other input data to be retrieved by the testbed provider to be able to execute;
  - *Return output data:* once an experiment completes, any results, such as performance monitoring, log files and so forth, need to be returned or at least alerted to the TEFIS data services.

The TEFIS data services provide support for all data needs from the external stakeholders, the *Experimenters* and *Testbed Providers*, as well as for the other TEFIS platform components. To establish what is required from the data services in terms of a metadata specifications as well as storage from the stakeholders, to begin with we need to go back to the three main scenarios included in the TEFIS project:

i. *eTravel*: this involves the compilation of a complex SOA-type eCommerce application, with a view to support the maintenance and ongoing extension of an online travel service. The use case involves three testbeds: ETICS for compilation, PACA Grid and PlanetLab for performance optimisation. The use case is important since successive stages of the test are dependent upon the successful completion of a previous stage.

ii. *IMS*: this involves an experiment covering a number of different stages in the lifecycle of a multimedia service. The use case involves the interaction of two test facilities: the BOTNIA Living Lab for end-user involvement, and the SQS IMS testbed to run and exercise the application itself. The use case is significant because it involves (a) the complete service lifecycle, but in addition (b) the co-ordinated execution of the application across several test environments.

iii. *eHealth*: the main part of the use case involves the secure transmission of large data packets, Patient Records, to be exchanged in *quasi* realtime to all members of a cross-disciplinary medical team. The use case is important in that it involves both functional (large data footprint

[2] This function is shared with the Experiment Manager, which currently allows aspects of experiment configuration and set-up to be searched on and re-used. Merging the capabilities within the Experiment Manager and the Experimental Data Interface will be addressed in connection with the new test cases. The Data Services will remain responsible for provided appropriate access control around objects within the file system, such that experimenters can publicise or hide their work from other users.

[3] An "application" in this context refers to the software object to be tested. This may be one or more executables to be compelled or stress-tested, or a web service requiring input from end-users.

requirement signfiicant bandwidth and speed) and non-functional (security and data protection) factors.

The three use cases are described in more detail from a requirements perspective in the following sections. To begin with, we consider the relationship between *tasks* and the testbed resources used to run the tasks within the test, before moving on in the subsequent sections to execution flow dependencies and resourcing, and finally consider what is needed after experiment design and provisioning for experiment execution.

4.1. The Three Initial[4] TEFIS Use cases

The three use cases are shown below, described in terms of the *tasks* the user wishes to run, and what is executed on the testbeds providing test facilities. For our purposes in this document, we define a *task* as an activity that the user performs – it is specified in terms the user understands and operates so that the user can get value from it. Tasks, therefore, can be seen as the high level logical stages in an experiment from the experimenter's perspective[5] and not computational processes. How tasks map to the execution environment turns out to be a significant consideration when defining the physical data structures used to support the stakeholders (see Chapter 7). In consequence, each use case is described in terms of the number of logical user tasks and how those tasks map to the execution environment.

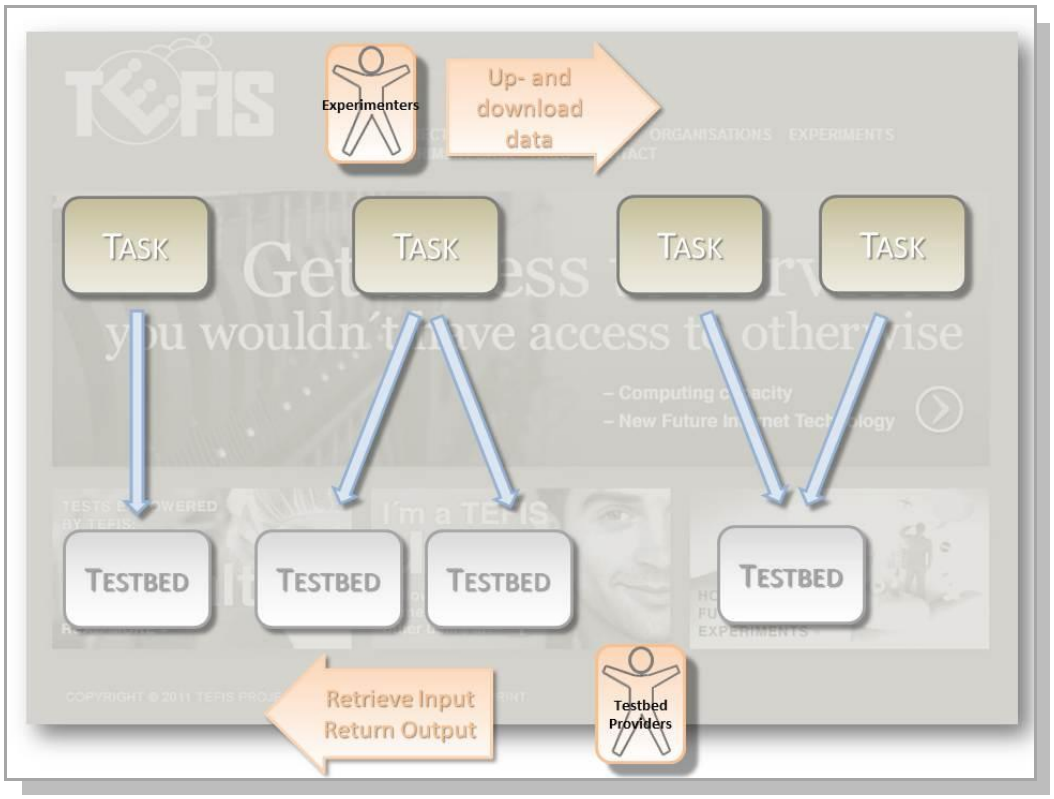

Figure 4: Generic Mappings task to testbed resource

[4] As a result of the *Open Call*, more experiments will need to be supported. At this time, there is insufficient information about these additional tests to factor into the discussion.

[5] For other TEFIS components, *tasks* may be viewed in terms of the steps involved in test execution.

It is possible to conceive of a number of different types of mapping between a task, for which the experimenter would wish to provide input and retrieve results, and testbed resource for which the testbed provider needs input and from which (s)he will return output. Figure 4 presents the main cases which are described below:

i. task and testbed resource are mapped one-to-one;

ii. a single task needs resources from multiple testbeds; and

iii. multiple tasks run on a single testbed resource.

In each case, the main issue for the data services is where data should be stored and/or written to. For a one-to-one mapping, this is simple: the experimenter writes input information to a single location, and the testbed reads it from a single location; similarly with the output data: task and testbed in- and output data are easily mapped. For a single task to many testbeds, then the experimenter still uses a single location associated with the task, but there are multiple testbeds: input must be retrieved by individual testbeds, and their output collated somehow as corresponding to a single task. The reverse is true when multiple tasks run on a single testbed. We will bear this in mind for each use case in the sections below. The issues for the Data Services can be summarised as follows:

Tasks to Resources	**Input Data**	**Output Data**
One-to-one	Input data are written to a single location, and can be retrieved from a single location.	Output data are written to a single location, and can be retrieved from a single location.
One-to-many	Input data are written to a single location, and can be retrieved from that location. However, different testbeds may need to access that single location. *Issue*: access control (multiple users reading from a single location).	Output data available from multiple sources. They may be written to multiple locations; *Issue*: references need to be provided to link all such locations. Alternatively, they may be written to a single, shared location; *Issue*: access control.
Many-to-one	Input data are written to multiple locations. *Issue*: a reference needs to be provided to link all such locations, so that the single testbed can retrieve all relevant input. Alternatively, all input is written to a single, shared location. *Issue:* access control (multiple users writing to single location).	Output data are written to a single location, and can be retrieved from that single location. *Issue*: access control (multiple users reading from a single location).

4.1.1. The eTravel Use Case

The eTravel use case relates to an experiment in two main parts: first, the optimal compilation of different services from an SOA environment, along with associated log and data files; and second, the performance of the compiled application across different execution contexts to establish the appropriate environment to run the eTravel application.

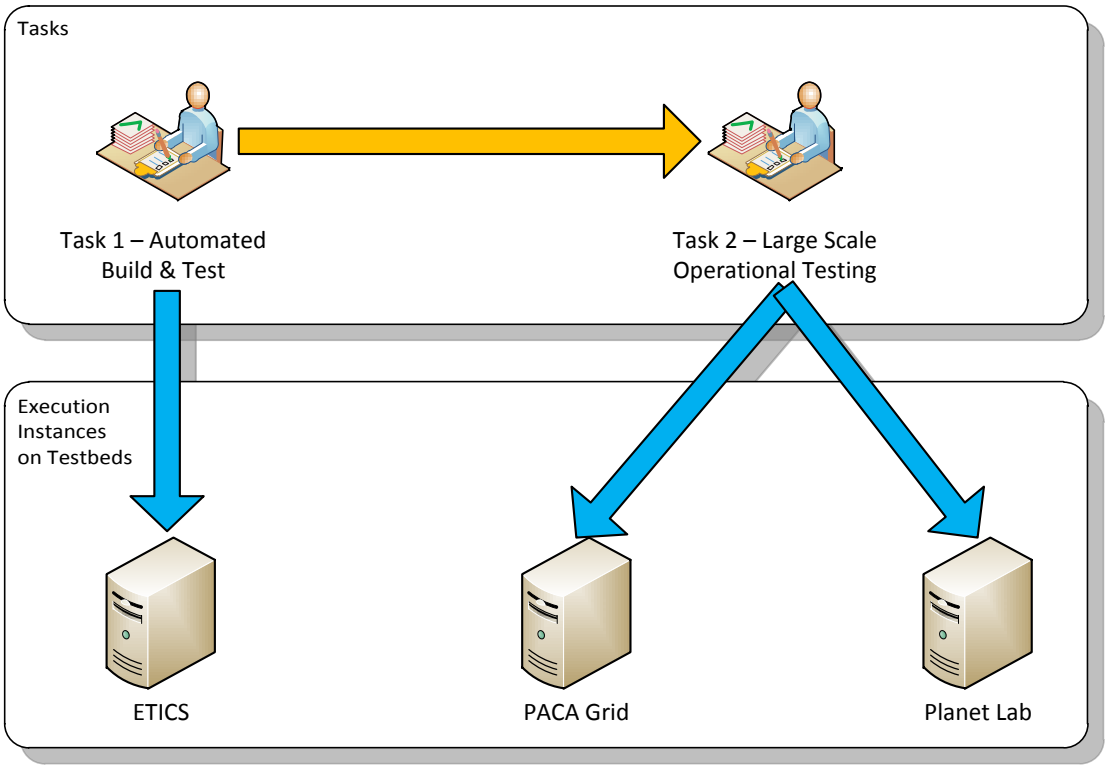

Figure 5: eTravel Use Case

Figure 5 shows the eTravel use case. Here we have the two tasks - the compilation of the application, and the exploration of the operational environment – run on different testbeds. The first runs on ETICS; and the second task on a combination of Planet Lab and PACA Grid. Tasks from the experimenter's perspective are separate from the testbeds: there is no consistent one-to-one relationship between task and testbed resource to run the test. This use case illustrates [i] one-to-one task to testbed, as well as [ii] one task to multiple testbed mappings. In the latter case, the data services must cater for multiple providers accessing and returning information for a single task.

4.1.2. The IMS Use Case

The IMS use case is shown in Figure 6, with the different stages in the experiment lifecycle indicated: Task 1 - testing service function (*Utility Testing*); Task 2 - the initial application tests (*Functional and User Testing*); and Task 3 - finally Business Model Evaluation.

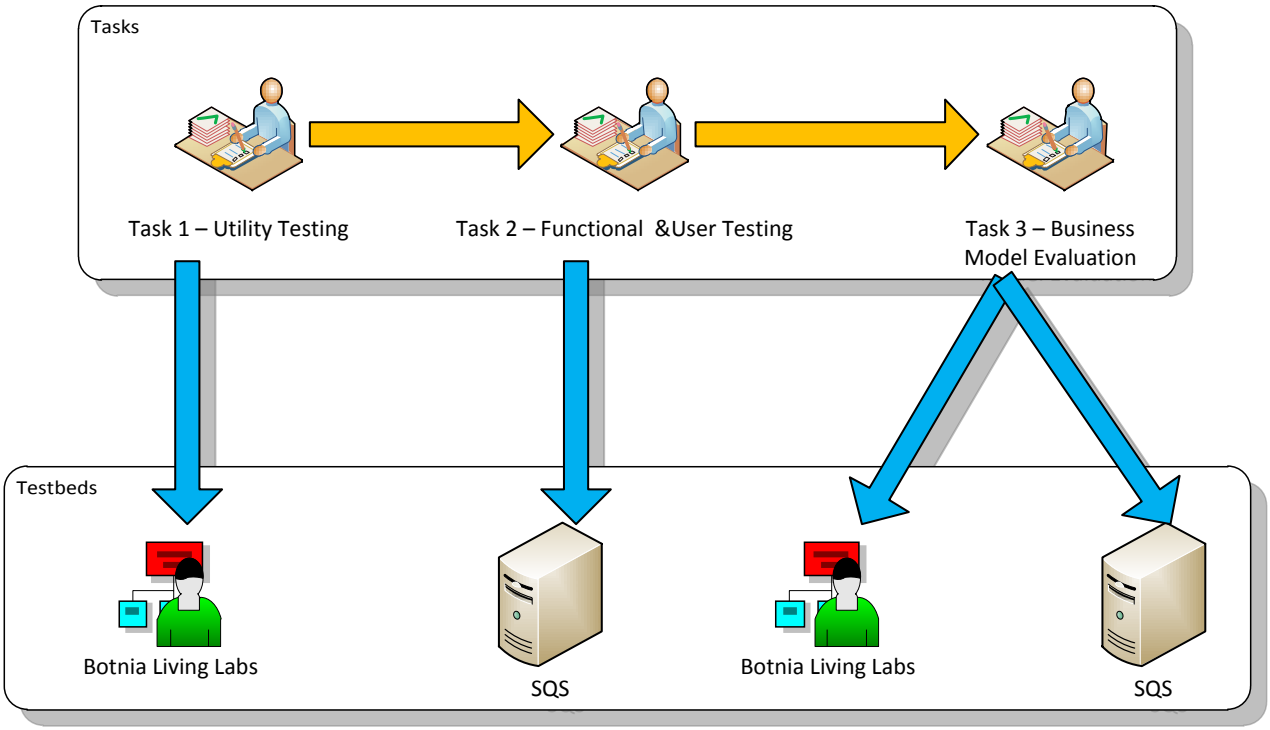

Figure 6: IMS Use Case

The first task runs using BOTNIA Living Labs, with subjects invited to consider what is being provided in the application; the second uses SQS to test the application runs as designed; and the third uses both BOTNIA and SQS, with users from BOTNIA Living Labs actually engaging with and trialling the multimedia application for the exchange of content in a mobile environment.

The use case illustrates [i] one-to-one task to testbed, as well as [ii] one task to multiple testbed mappings. Like the eTravel case (see Section 4.1.1 above), the data services must cater both for the simple one-to-one task to testbed mapping, but also for allowing multiple providers to access and return information for a single task.

4.1.3. The eHealth Use Case

The third of the initial TEFIS test scenarios involves an eHealth application. There are four basic tasks involved in the assessment and care of a patient: the patient presents themselves to the medical services either to a doctor or via paramedics (*presentation*); the doctor (or paramedics) need to make a *diagnosis* and then *treatment* planned and put into place; and finally the patient is re-assessed and *discharged*. At all stages, medical records are exchanged, consulted and updated.

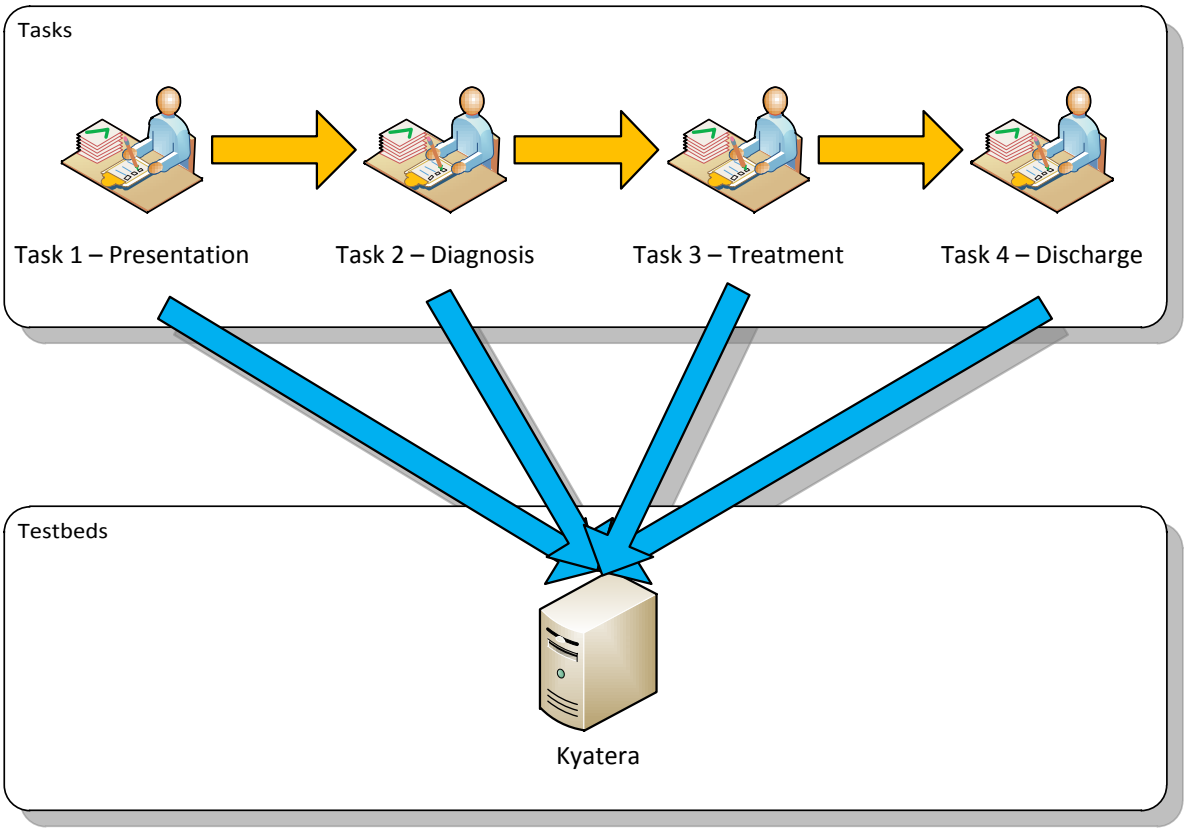

Figure 7: eHealth Use Case

Figure 7 shows the eHealth use case. Here all four tasks run on the same testbed[6]: this use case therefore illustrates [iii] multiple tasks running on a single testbed. In this final case, the TEFIS data services must maintain some degree of separation between the user tasks, and the testbed provider will access and return information to those multiple tasks.

4.2. Tasks and Execution Instances

In relating the tasks that users run to what is actually running on the testbeds, we have seen that there is a significant difference between the activities the experimenters need to do and the actual instance(s) used to support those activities on a testbed. For example, activity executed by the user may seem to them to be one single process, but may be running on many testbeds behind the scenes, as seen in both the multimedia services (IMS) and the eTravel use cases. Alternatively, multiple and separate user tasks may in fact be run on the same testbed resources, such as the eHealth use case. Therefore it becomes necessary to separate the user's view of a task from its resourcing and execution. To this end, we introduce two new concepts: the *Task* and the *Execution Instance*.

- A *Task* is a logical activity in the experimenter's domain – it is an activity the experimenter understands and can get value from.

[6] Subsequent iterations of the experiment may involve the introduction of other tasks running in different test environments, such as build and test on ETICS and computer-intensive diagnostic analytical techniques run on PACA Grid, for instance.

- An *Execution Instance* (EI) is a single execution at a testbed, for example one execution of an application with a certain parameter set, or a user-acceptance test or survey in a living lab. The EI is the actual hardware, software and human resources supplied by the testbeds in action to address the needs of the user.

The relationship between tasks and execution instances can vary depending on the situation (Figure 4). To summarise the initial TEFIS use cases above:

i	One *task* can map to one *execution instance*	eTravel (Figure 5); IMS (Figure 6)
ii	One *task* can map to many *execution instances*	eTravel (Figure 5); IMS (Figure 6)
iii	Many *tasks* can map to a single *execution instance*	eHealth (Figure 7)

Therefore we have to provide a means to be able to store Tasks and EI's separately. The mapping between the tasks that a user executes and the actual executions on the testbeds is defined by the experimenter or the testbed provider, or both, and is outside the scope of data management; what the Data Services need to provide is

- a means for the independent storage of Tasks and EI's;
- together with a mechanism for relating them to each other.

The proposed locations for *task* and *EI* storage are presented in Chapter 7. But first, we examine the various requirements in the following sections for experiment handling, including the dependencies between tasks and EI's. For now, it is important to remember that *task* and *execution instance* are both important concepts within the data model underlying the Data Services. These have been developed and defined as implementation after the initial metadata specifications were outline [1] and in light of the actual running of the target test scenarios.

4.3. Dependencies

The analysis of the initial TEFIS use cases mentioned earlier also showed that there is often a logical order for tasks – the sequence that they should be executed in. It makes sense to order the tasks in a particular way, but also there may be dependencies between tasks (e.g. one task's output is required as a subsequent task's input). By the same token, there can be dependencies between Tasks and EI's. These dependencies can take different forms. For example, one task cannot begin until another has finished, or an EI cannot begin until another has started. Finally, there are often data dependencies – for example one task or EI needs the output of another task or EI as its input data. We summarise the nature of these dependencies using the following terms to describe the two basic types:

- End to Start: one task must have completed before the next begins. This would be the case, for instance, when the second task requires the results of the first task; and
- Start to Start: two tasks must start together. This would be the case when one task is feeding the other with information or input. For example, a web service under test by real end-users must be started together with the end-users being ready to dial in or connect with that service.

The following table introduces the dependencies between tasks and execution instances. In each case, the dependency may either by *End to Start* or *Start to Start*, in respect of whether the one (task or execution instance) must have completed before the next begins, or not.

Table 1: Dependencies between Tasks and Execution Instances

Dependency between		Description	Example(s)
Task	Task	One task may need the results of another task before it can begin One task may set up the necessary conditions for a subsequent task	Input of one task is the output from a previous task The first task may be the configuration for a following task
Task	EI	An EI may not be able to run until certain steps have been taken	The task may be necessary preparatory steps, such as recruitment of Living Lab volunteers to evaluate an application
EI	EI	One EI cannot begin until another has begun or completed	User evaluation by Living Lab recruits cannot begin until the application to be evaluated has been deployed and made available

In the case of eHealth, the dependencies are clear: each of the tasks *presentation, diagnosis, treatment* and *discharge* (Figure 7) requires the previous task to have been started if not completed before it can run successfully. A patient must be known to the MDT (that is *presentation* have completed) before any *diagnosis* can be attempted; and so forth. The tasks must necessarily execute in a given order. The testbed then provides the necessary capabilities to be able to store, secure and make available the appropriate patient records at any time during the task sequence.

Dependencies for the IMS and the eTravel cases needs a little more attention, and are described in the following sections.

4.3.1. Dependencies for the IMS Use Case

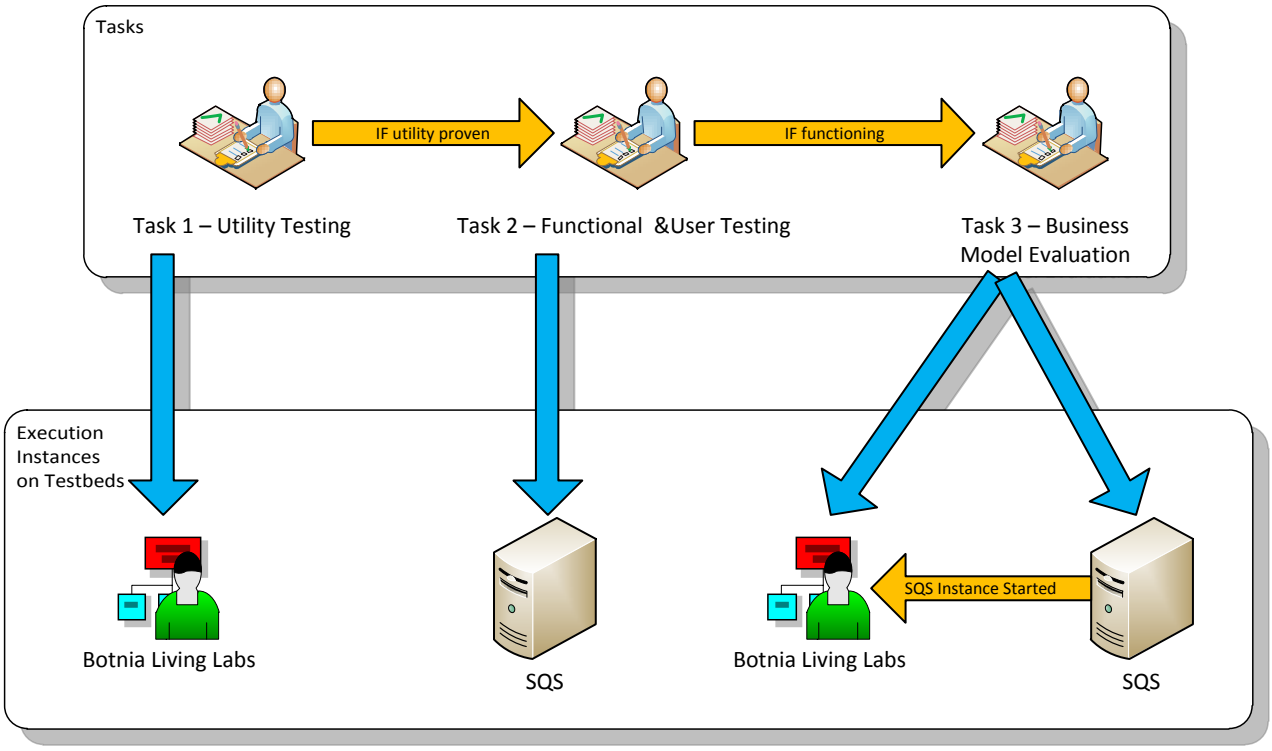

Figure 8: Some IMS Use Case Dependencies

The main dependencies for the IMS use case are shown in Figure 8. There is definite sequence for the tasks: the utility testing (Task 1) must show that the application under test is useful before the functional & user testing (Task 2) can begin, and only when this is successfully complete, the business model evaluation (Task 3) can begin. Here we see that the sequence of tasks is necessary, as each task must finish and return a confirmation that the task's tests have been successful before the subsequent task can begin. We call this type of dependency "*End to Start*", meaning that one task must *end* before the subsequent tasks *starts*.

There is an additional dependency in Figure 8, and this is between two Execution Instances. In Task 3, there are two EI's, one on BOTNIA and the other on SQS. It is a requirement of the experiment use case that the two EI's can talk together, i.e. the survey running in BOTNIA can provide user feedback back to SQS and interim SQS output can be seen by the test subjects. This means that the two EI's must run concurrently. We can use another type of dependency to reflect this concurrency – in the figure, the BOTNIA EI cannot begin until the SQS EI has started, and the application deployed and running. Therefore we have a "*Start to Start*" dependency – the dependent task begins when the independent task has begun. (We could have illustrated this example by having the dependency the other way round, i.e. SQS depending on BOTNIA starting, but since the SQS task is computer-based and the BOTNIA task involves human subjects, it makes sense to start the computer EI first so the human subjects are not held up by waiting for the computer EI to start.)

4.3.2. Dependencies for the eTravel Use Case

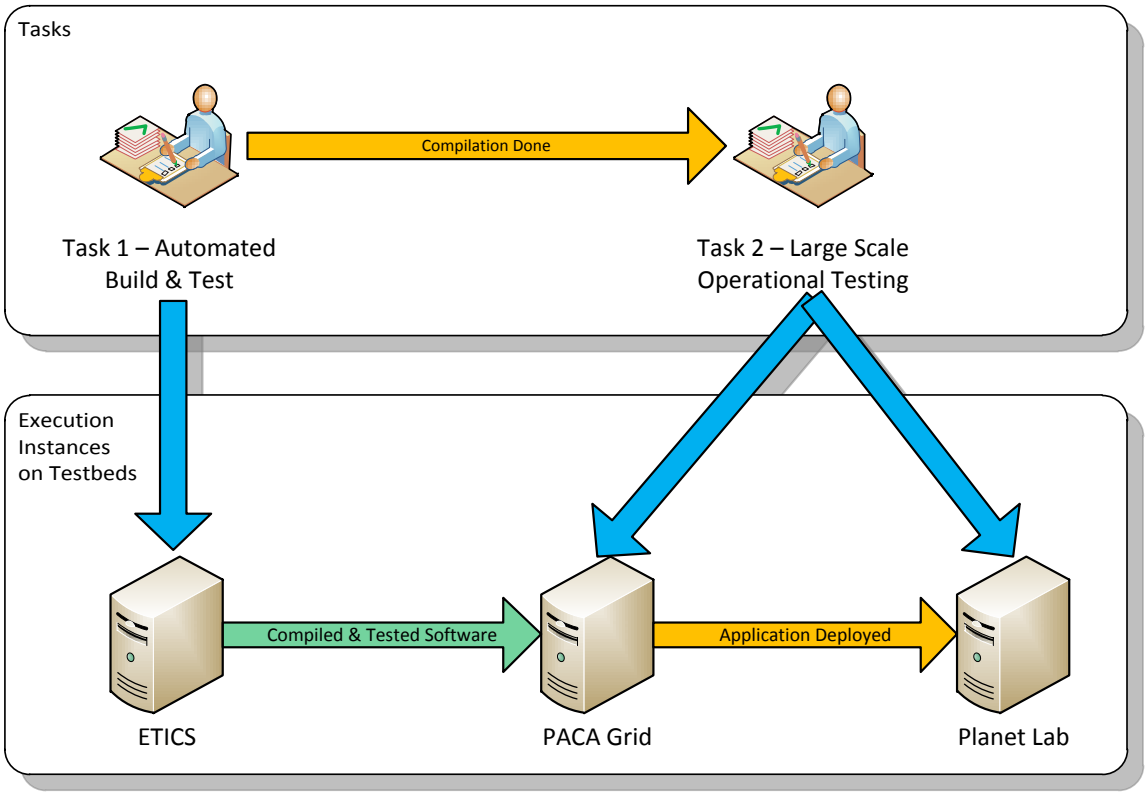

Figure 9: eTravel Use Case Dependencies

Data dependencies (along with other dependencies) are illustrated in the eTravel use case, as shown in Figure 9. As with the IMS use case, there are dependencies between Tasks and EI's. Task 2 (large scale operational testing) requires that Task 1 (build & test) is complete before it can start: an *End-to-Start* dependency. There is also a data dependency between ETICS (where the building and testing takes place) and the deployment of the built and tested software at PACA Grid (illustrated by the green arrow): an *End-to-Start* EI dependency. Also similar to the IMS use case is the requirement for two EI's to run concurrently – at PACA Grid and Planet Lab. This is because PACA Grid hosts the nodes for running the large scale operational tests on, and Planet Lab provides management of these nodes[7]. Once the software is deployed at PACA Grid, the Planet Lab management of the PACA Grid nodes can begin, and so there is a dependency between PACA Grid and Planet Lab: a *Start-to-Start* EI dependency.

4.4. Workflow

In the preceding section, we have talked about dependencies in terms of tasks as well as execution, and have identified different types: *End-to-Start* where one task or EI cannot start before the preceding one has completed, or *Start-to-Start* where one task or EI cannot start before a related one has been started. These dependencies were largely based on the results of one task being needed by a subsequent task (for IMS and eTravel). For the eHealth case, we saw that the only real dependencies were in the sequence in which tasks were executed. We need now to consider how the experimenter might define

[7] An additional experiment definition has been proposed for eTravel, whereby PACA Grid is used for performance testing, and PlanetLab independently for network tests. In this case, the dependency would be End to Start, since the performance testing would beneficially need to be complete so that detailed configuration information is available for the network tests.

an logical order in the execution of tasks, bringing together any sequencing requirements as well as data dependencies.

The sequence of tasks and the dependencies between them comprise a *workflow* that describes in the user's terms the order of the tasks. The workflow can be built by the experimenter or made by a testbed provider, but we assert that the workflow should always contain user-focused tasks (as opposed to testbed-focused execution instances) because the workflow will be executed by the experimenter and the experimenter is concerned more with tasks, and not how they are executed[8]. This implies, of course, that EI-to-EI dependencies, and to some degree task-to-EI dependencies, will not be captured in the workflow as defined here from a user perspective. The Experiment Manager will initially assume responsibility for the mapping between *task* and *execution instance,* though the experimenter themselves will also have the option to specify *execution instances.* In addition, a testbed resource perspective will need to be generated as part of the Workflow Scheduler or via the TEFIS Connector Interface (TCI) either at runtime. This is beyond the scope of the Data Services.

At any event, in defining the sequence of tasks to be executed and the dependencies between them, the workflow becomes the "program" for executing tasks (because it determines task order and dependencies). As developed in the Experiment Manager (*q.v*. [5]), it captures this information as well as an indication of the resources involved for the EI or EI's used to handle the tasks listed. As far as the Data Services are concerned, because the workflow is like a program, we need to identify each run of a workflow, so we can see what data it used and produced, and which resources it consumed. To represent this, we use the concept of a *test run* – a particular run of a test.

Workflows, therefore, capture the experimenter's perspective of how a specific instance of an experiment, the test run, is defined. This includes the order of tasks as well as any dependencies between the tasks, and some reference to the resources to be used to support task execution. The workflow is generated by the user with the aid of the Experiment Manager. However, other components, such as the Experiment Manager, as stated above, is responsible for mapping the tasks defined in this workflow to the EI's needed to satisfy them. For the Data Services, the *workflow* therefore is the concept which identifies the experimenter's wishes in terms of experiment steps (*tasks*), the dependencies between the steps and the data associated with them, and the *test run* is an individual instance or execution of that *workflow*.

4.5. Resourcing

As well as the *workflow*, the Data Services need to engage with the resources used to support any specific *test run*. There are different resource types in TEFIS – some are computer hardware and software based, and others non-ICT based. Of the six original TEFIS testbeds, five provide ICT resource: ETICS, Kyatera, PACA Grid, PlanetLanb and SQS; whilst the sixth, BOTNIA, provides non-ICT resource – experiment design expertise and consultancy skills as well as test subjects. Whatever the resource type, when an experiment runs on a testbed, it consumes testbed resource, so records need to be made of the resourcing requirements, which resources are reserved for which experiment, and how much resource is consumed by experiments. For the Data Services, the type of resource is less important than who owns

[8] Experimenters may be concerned with the way a task is executed, if that is the purpose of the experimental investigation. The distinction being made here is to highlight the difference between an experimenter defining what is done, and the TEFIS platform taking responsibility for executing it.

or controls that resource: the Data Services[9], as mentioned previously (see Section 4.2), need to allow access to the appropriate locations to read input from or write output to.

In order to acquire testbed resources to enable them to run experiments, the experimenter has to procure them from the testbed provider. During procurement, three different aspects of resource specification information are needed at different times. These aspects correspond to different stages of the resource procurement process and reflect:

1. What resource the experimenter wants – this is the specification of the resources required for the experiment (the "shopping list" in the Virtual Customer Testbed (VCT)), and is handled with the support of the *Experiment Manager*;

2. What resource specification the testbed provider is offering (what is "for sale"); this occurs via the TEFIS *Portal* during or after a testbed registers with the TEFIS platform; and

3. What resourcing is agreed between the experimenter and testbed provider (what is "bought" by the user) – this is the outcome of negotiation[10] between the experimenter and testbed provider and represents the amount and quality of testbed resource that can be actually used by the experimenter.

As indicated above, the outcome of the resource procurement process is a *resource booking*, and this enables the experimenter to run their experiment. It also provides limits on the experimenter's use of the testbed provider's resources. The *resource booking* is simply a record of what and how much resource is booked for the experimenter at a testbed for an experiment. There can be a number of different types of resource booking, depending on the characteristics of the resource, the business model employed by the testbed provider, and the purpose to which the resources may be put. Some examples are discussed next.

- A resource can be booked for a single execution of an application – this is a one-off operation, and the booking is similar to a ticket for a train journey, in that it cannot be used more than once.

- A resource can be booked for a particular time slot, for example a computing resource can be booked for a two-hour time slot beginning at some specified time. The resource can be used for multiple runs of applications or other purposes during this time slot. It is the time slot that is the important aspect of this booking type, not what the resource is used for. This is similar to a hotel room reservation, where the booking is time-slot based.

- Bookings may be on best-effort basis, meaning that the user will get the resource if it is available, but if not the user may have to wait. This booking type will likely be much cheaper than more

[9] The amount of resource is significant to other components, including the testbeds. For instance, when the resources are booked, the testbed provider needs to know how much to allocate. At runtime the amount of resource used needs to be tracked and reported for billing purposes.

[10] This is currently handled as a manual process between user and testbed provider. TEFIS will request, as stated in [1], the resource required for a specific experiment instance which is honoured by the testbed provider on this basis.

guaranteed booking types. There may be levels of best-effort resource bookings, for example a high-grade booking (which costs more than a low-grade booking) may have a shorter queue.

- Regular users may be allocated quotas of resource per unit of time, for example a user could be allocated 10GB of data upload to a storage service per month. This can be tied to charging policies, whereby the user gets charged extra if they go over their quota.
- Resource bookings may be combined – for example a software application has a ticket-based booking for one execution but runs on a platform as a service testbed that is booked to a two-week period. Here, the two bookings can be combined to provide the complete execution environment for an EI, and the need for both bookings can be represented in the Starting Conditions for an EI.

The specific booking methods implemented by TEFIS are beyond the scope of the Data Services. However, the *resource booking* will allow the Data Services to identify who is to be given access to data, and potentially for how long. Similarly, resource consumption falls outside the Data Services, except in terms of local (that is within the TEFIS domain) storage. The actual recording of resource consumption may be done by monitoring (see [2],[3]), and monitoring takes place during the execution of the experiment, discussed in the next section. For now, it is important from a Data Services' perspective to be able to handle and act on a *resource booking*; this concept will be discussed further in Chapter 5.

4.6. Experiment Execution

Once the experiment has been designed and resources have been procured to enable its execution, the experiment may be run. The main requirement here on the Data Services is to provide a means of storing experimental data (covered in the deliverables [6] for local and [7] for remote storage) and provide a metadata schema that can enable:

- Identification of data items, for example which experiment a data item belongs to;
- Recording of the relationships between data items, for example a data item might be an output to a related input data set;
- Recording the provenance of data items, for example we should know if and when a data item is written to, and by whom or what; and
- Representation of monitoring data for the recording of resource consumption and performance.

These requirements form the basis, along with concepts introduced in the previous sections within this chapter for what is described in the following chapters. Chapters 6 and 7 consider the metadata and physical storage structures respectively associated with the implementation of Data Services in support of TEFIS experiments. To begin with, Chapter 5 considers the experimental lifecycle as verified in connection with the initial TEFIS use cases.

5. Experiment Lifecycle and Data models

In the initial *Specifications of Experimental Metadata* (Chapter 5, [1]), a generic experimental lifecycle was introduced with individual steps, including: *Plan, Request, Provision, Deploy, Run, Evaluate, Re-run,*

Collate. For any given experiment or instance of an experiment, these may be grouped together from a user's perspective as: *Design* for the planning stage; *Provisioning* to request and provision the resource; and *Execute*, covering deployment, running, evaluation and potential re-run. Design, provision and execution become the main emphasis, therefore, with the individual steps in the original and overall lifecycle sub-tasks within those steps. What remains is the collation of the data and its publication (making it available to others): the curation of the experimental data, which involves access control measures and retention, both of which need to be handled separately and in collaboration with other TEFIS components. For the TEFIS Data Services then, we need to focus specifically on the three main stages to ensure that suitable support is available for all main steps as well as sub-tasks.

In consequence, based on the use case analysis and requirements above (Chapter 4), we can determine an initial and pragmatic experiment lifecycle for the purposes of data management:

1. *Experiment Design*: The Experimenter designs their Experiment. Here the Tasks (see Section 4.2), the Workflow (Section 4.4) and Dependencies (Section 4.3) are determined, together with the requirements for Resources on the Testbeds (see Section 4.5). This stage of the experimental lifecycle is modelled in Section 6.3.1.

2. *Resource Procurement* (Experiment Resourcing): The Experimenter procures Resources to run their Experiment. Here the Resources required for the Experiment are matched with Resources provided by Testbed operators to create *Resource Bookings* that enable the Experimenter to run their Experiment (Section 4.5). Section 6.3.2 covers this stage of the experiment lifecycle.

3. *Experiment Execution*: The Experimenter runs their Experiment. Here the Execution Instances are deployed (or existing ones used) on Testbeds to run the Experiment. The running of the Experiment generates and uses Test Data. This is covered by the model in Section 6.3.3.

To address this lifecycle, we have created three Metadata models, one for each stage of the lifecycle, as discussed in Chapter 6. We also present a physical data storage model in Chapter 7. We have metadata and physical models because it was that the data model from [1] really involved two separate categories of data:

a. the physical storage of experiment data; and

b. metadata that describes the experiment data.

During the early phases of implementation, it became apparent that the two types of data need to be dealt with individually. This was mainly because of the need to store experimental data in the local TEFIS data store (the RPRS) and/or the remote data store (the TIDS). This relates to the physical aspect of data services. But in addition, this derives from a need to be able to search and find the stored using its metadata attributes (the descriptive side to data management). The rationale for both the physical storage and the metadata are summarised below, along with the requirements previously identified (Table 9, [1]).

The main purposes for physical storage are to:

provide a secure storage facility for experimental test data	4[11], 7, 8, 19, 21
enable access for the experimenter to their data;	7, 13, 20
give the experimenter control over which testbeds can access which data;	7, 8, 13, 21
provide remote access to experimental data for authorised users.	8, 19, 21

The main purposes of metadata are to:

enable users to find data items quickly and easily, using different types of search terms (e.g. search using experiment owner, description, keywords, testbed provider, date);	7, 12, 14, 15, 19
provide logical structure so that data elements may be related to each other, for example it is important to know which experiment a data item belongs to;	14, 19
record audit events on data so that its provenance is known (e.g. who changed data, when and how).	12, 14, 15, 17, 19, 20

The main stage of the lifecycle for these requirements on the Data Services is during Experiment Execution[12], when preparatory data need to be stored in an appropriate location, as well as output written to an agreed location. Requirements here are satisfied mainly through the physical storage of experimental data. However, the design and procurement lifecycle stages require information *about* the experimental data: where they came from, how they were produced, who owns them, and so forth. These stages of the lifecycle are better catered for through requirements associated with metadata. To begin with, then, Chapter 6 covers this aspect of the data: the metadata models to allow search and data management. Following then, Chapter 7 focuses more particularly on the physical models. Both chapters also show how the two models relate to each other, and how they satisfy the requirements from the use cases.

6. Metadata Models

This section discusses the metadata related to the experimental lifecycle stages: experiment design, resourcing and execution. A series of related and overlapping data models are presented showing the concepts and their relationships in the three areas above. These data models form the basis of a

[11] These are the requirements on the Data Services outlined first in [1]; and reproduced as Table 3 in Section 11 – Appendix II, below.

[12] In addition, there may be a requirement to provision an amount of resource (storage, for instance) to be allocated for the running of the experiment from a TEFIS' perspective. Some components (during experiment definition) use caches of existing information; others, such as Monitoring, assume initially that data will be cached at the testbed location, and can only request storage capacity once the experiment is in progress, if not complete.

metadata schema that may be used to describe physical items like test data files and folders, so that they can be managed from different points of view. In a later section we will show how the metadata models are mapped onto physical items.

Our aim with data models is to provide a metadata schema and structure that may be used by other components in TEFIS for recording information. The intention is not to be over-prescriptive with these data models but to remain flexible about their use – for example project partners can use the fields supplied and add to them as they see fit. Having said this, we think it is important to define the concepts in the areas of experiment design, resourcing and execution, and to show how they relate to each other. This will provide project partners with a data dictionary for concepts and contexts for each concept in terms of other concepts.

6.1. Metadata Standards

A number of different metadata standards associated with experimental and scientific research have been proposed[13]. The need for the standardisation of these metadata specifications has been recognised for some time. Much of the motivation rests on searching the increasing range of scientific data:

> As the sea of information grows, *being able to locate, discover, link to, search on, re-purpose, integrate, track, exchange, or sell a given information resource* all tend to become more complex processes. Good metadata practices reduce some of this complexity and help publishers harness the new opportunities that new technologies will bring (p13, [8]) [*Our italics*]

Much progress has indeed been made, it seems, with Dublin Core (see Section 6.1.1) providing an effective and simple method of cataloguing published data, for instance. It is not altogether clear, though, that it has become easier to search and retrieve technical data without specialised knowledge of the domain.

> Most metadata standards have focused on the descriptive elements needed for discovery, identification, and retrieval. […] Technical metadata is one area that still does not get much attention in metadata schemas. The *effective exchange and use of digital objects described by the metadata often requires knowledge of specific technical aspects of the objects beyond its filename and type*. (p12, [9]) [*Our italics*]

This has led in turn to the development of a framework to allow for different standards to be accessed, but also in a way that more intuitive and without the requirement for specific technical knowledge:

> We believe the [Metacat] framework can help facilitate new methods for data sharing and accelerate the pace of scientific discovery by granting researchers access to rapidly growing data stores that can complement and enrich their analytical insights. (p68 [12])

In this section, against this background of wanting to search and access technical data, but also in an intuitive way which does not require extensive domain knowledge, we briefly review three well-known proposals for metadata standards for use with scientific data: Dublin Core (Section 6.1.1), The Common Scientific Metadata Model (CSMD) (Section 6.1.2) and the Full Metadata Format (FMF) (Section 0). Finally, we introduce the proposed TEFIS Data Services metadata schema (Section 6.1.4), which is set out

[13] See the review in [9], for instance.

in some detail in Chapter 10 Appendix I: *Metadata Schema*. This latter, TEFIS schema has been developed on the basis of what has been derived from the existing standards, and in consideration of the strengths of any individual standard.

6.1.1. Dublin Core[14]

The Dublin Core Metadata Initiative (DCMI) began in 1995 as a result of a joint effort between publishing, library services and the academic community. By 2001, it had become a NISO (National Information Standards Organization) standard, and an ISO (International Organization for Standardization) standard by 2003. There are 15 (from an original 13) elements[15] covering identity, ownership and rights, and provenance:

> *Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, Rights*

Each element is described in terms of its intended use, along with recommendations for the type and format of the value to be associated with the DCMI element. It is very simple, widely used and can be extended. Most standards reference or are derivative of the Dublin Core specification.

The fifteen core elements provide adequate coverage for most eventualities. Its simplicity and flexibility makes it very powerful. However, the lack of particular constraints on the values for each element mean that it can be difficult to provide anything other than a simple search. The metadata schema for TEFIS should similarly be *simple, extensible* and support easy, user-centred *searches*.

6.1.2. Core Scientific Metadata Model

Core Scientific Metadata Model (CSMD) was developed over 8 years within the Science and Technology Facilities Council. Individual classes within the model provide references to locate the actual data items or contents. The main classes include:

Investigation	This is the basic unit in the model, and has attributes such as *title, abstract, dates* and *unique identifiers* providing links to the original study, as well as the *facility* (test resource) and *instrument* used to collect the data. The attributes echo some of the core elements from Dublin Core (q.v. Section 6.1.1)
Investigator	This describes those who took part in the study, along with their specific roles in the study as well as the institution they are affiliated to.
Topic and Keyword	These are controlled and uncontrolled vocabularies to annotate and index the investigation.
Publication	These are any published works associated with the data and/or study.

[14] See: http://dublincore.org
[15] http://dublincore.org/documents/dces/

Sample	This element relates to the material under investigation.
Dataset	Any individual datasets for the current investigation, including individual test runs or analyses.
Datafile	Datasets may contain nested datasets. Each datafile contains more detailed information, including name, version, location, format and so forth.
Parameter	These are specific values associated with the investigation, such as temperature, pressure etc.
Authorisation	Records access permissions.

The intention is to allow for work to be searched and found, with a view to referencing the work, validating results or creating derivative work. Curation and retention is less well served. The CSMD, however, does allow *cross-referencing* between related and derived work.

6.1.3. Full-Metadata Format

The Full-Metadata Format (FMF) is intended to provide a portable markup for scientific and technical data sets. The intention is for the datasets so marked to be:

Readable and self-documenting	there should be no specific requirement for specialised tools to be able to read and processed.
Flexible but structured	although there is a specific suggested format (the metadata fields occur at the beginning of a dataset before the raw data themselves), there should nonetheless be some flexibility in what is contained and how it is formatted.
Fail-safe and compatible	datasets should be able to be processed in all possible environments and without the danger of obsolescence.
Searchable	the data (including the metadata) should be able to be searched and accessed.

To some degree, these are common, if not explicitly stated, aims for each of the metadata standards. But the FMF does emphasise some important non-functional aspects of data handling and storage:

i. Cleat-text documentation of scientific data simplifies its re-use

ii. Using plain text files makes the data ideal for long term preservation

iii. Communication of the data does not need a complex infrastructure; text files can be sent by eMail or even be printed to analogue media

iv. Because the data is connected to the relevant units (ie. quantities), special software which is able to process these units can be used

v. Use of relevant units enables a semantic search within a collection of data sets

Interestingly, an *End-of-Line* (EOL) character in the simple text-based files can make portability across operating systems problematic[16]. FMF is, though, a useful approach. It separates markup from the raw data into different portions of the data store. Markup and data are therefore *quasi* independent. The most significant feature of the FMF, though, is *simplicity* and *portability.*

6.1.4. The TEFIS Data Services Approach

The metadata markup and schemata suggested here have provided the basis upon which a TEFIS Data Services metadata schema has been developed (see Chapter 10 Appendix I: *Metadata Schema*). As with CSMD (Section 6.1.2) and FMF (6.1.3) much of the schema is derivative of and related to the Dublin Core (Section 6.1.1) as indicated in the Appendix below. The intention has been to retain the *simplicity* and *extensibility* of the Dublin Core, whilst introducing additional *cross-referencing* elements similar to CSMD. As an XML-type schema, it would be similarly portable and simple to interpret as FMF. The remainder of the elements in the TEFIS Data Services schema reflect the needs of the TEFIS operator and TEFIS testbed provider stakeholders, to allow them to identify specific elements and datasets appropriate for running individual instances of an experiment as well as reporting back the status and results of those runs.

In all, the proposed schema supports user as well as operator (or component) initiated searches, which is one of the main motivations for many of the metadata schemata that have been developed. However, as highlighted under the brief introduction to the Dublin Core (Section 6.1.1), not constraining the value fields for schema elements can make searching problematic. CSMD attempts to get round this problem with the introduction of the *Topic* and *Keyword* elements, one with, the other without a controlled vocabulary. Searches will tend, under those circumstances, to be easier and therefore encouraged across those items within a controlled vocabulary. In consequence, there will be a need to provide a well understood and familiar controlled vocabulary: users may well be expected to know more about the field than expected. This could potentially leave the approach open to the criticism that users need to know "specific technical aspects of the objects beyond its filename and type." ([9], *loc.cit*., and see Section 6.1).

One of the main drivers for the TEFIS platform is the ability to support users (experimenters) on *their* terms: they may be well versed in their own field, but not particularly experienced with resource definitions; hence the support provided by the Experiment Manager to define experiments in an intuitive manner ([5]). We have sought to avoid this problem in the TEFIS Data Services by offering experimenters the ability to search explicitly against the metadata terms and associated controlled vocabularies as well as to run free-format text searches in a similar fashion to a simple *Google™* text-based search: the experimenter can look for related work based on text descriptions or sets of collocated keywords in terms they understand, rather than as fixed by the metadata schema[17].

[16] Though this is really just a matter of knowing in advance to set appropriate parameters on any up- or download between operating systems.

[17] This is introduced with the *Experimental Data Interface* in [5]. Implementation detail will be provided in follow up deliverables.

Notwithstanding the related search capabilities, the metadata schema proposed for the TEFIS Data Services includes:

Experiment Info	This includes information elements related to the experiment itself, its source, owner, type and dates created. It is similar to the *Dublin Core* identifier elements.
Input Data	These are data related to experiment set-up, including application and environment configurations.
Output Data	These are data elements which were produced during a specific test run, including monitoring output.
Experiment Profile	This is the identifier for the generic context of an experiment (owner, description, status; similar to *Dublin Core*) and related works (the cross-referencing concepts of CSMD).
User Information	This is a *Dublin Core* derivative relating specifically to the owner to and contributor to an experiment (cf. the CSMD *Investigator* element).
iRODS tags	These are internal TEFIS Data Services elements for TEFIS components to query directly against the data management system[18].

The proposed metadata schema is not meant to be rigid and prescriptive. Instead, it offers the most appropriate features from the related standards discussed in this Chapter, along with specific details in support of the three main TEFIS stakeholders.

6.2. Data Concepts

This section gathers together all the terms defined so far and presents them with an official TEFIS definition. These are the concepts which need to be catered for in our models at some level either directly (they are included with the models, such as *Workflow, Task* and *Execution Instance*) or indirectly (concepts that are relevant within the context of TEFIS, though not specifically to be included in the models, such as *TEFIS Portal, TIDS* and *SLA*). The list in the table below is an update of the data

[18] As outlined in Chapter 11 Appendix II: *Technology Choice*, the TEFIS Data Services is based on the open source iRODS solution.

dictionary presented in [1] and represents modifications and additions made as a result of a greater understanding and insight obtained as the project has progressed.

Table 2: Main classes within the logical data model

Class	Description
Application	This is a component of the overall Experiment, and typically refers to a self-contained piece of functionality run on a testbed for experimentation purposes. An application could be software executables that run on testbeds, or be self-contained tests run on living labs. In the IMS case, for instance, it could be a mobile phone app that the experimenter wants to test.
Contract[19]	The legal agreement between experimenter and TEFIS. Its purpose is to determine the basic terms and conditions of the testing service provided by TEFIS and the commitments on both sides of the agreement. The *contract* is separate from an *SLA*, because it contains more general terms and is not test-specific. As such, the *contract* can be negotiated once per business relationship between experimenter and TEFIS, but referenced each time an experiment-specific *SLA* is created.
Dependency	A Dependency is a constraint that means one *Task* or *EI* needs something from another entity (commonly another *Task* or *EI*) before it can begin executing. For example an *EI* needs output from a preceding *Task* before the *EI* can start (see Section 4.3). Dependencies are implemented as *Starting Conditions*.
Execution Instance	An Execution Instance (EI) is a single execution at a *Testbed*, for example an *Application's* execution; under certain circumstances, it may represent a long-term rental of some storage or allocation of resource. Execution Instances are related to *Tasks* in two ways – there may be multiple *Tasks* per Execution Instance (e.g. a long-lived EI), and there may be multiple EI's per *Task* (e.g. hiding complexity of multiple executions inside a single *Task*). It is the decision of the *Experimenter* or the *Testbed Provider* to determine the exact mapping between a *Task* and an EI. Also, because a *Test Run* is an instance of a Workflow, a single *Test Run* may contain multiple Execution Instances. See Section 4.2.

[19] These classes have been retained from original concepts in [1], but are not developed further in the current deliverable.

Class	Description
Experiment	The Experiment is the main record of testing in the RPRS. It represents the grouping of related tests with a common goal, for example to establish the validity of a hypothesis. As such, the Experiment behaves like a "parent folder" for multiple *Test Runs*, *Execution Instances* and *Tasks*, which represent individual sub-components of an Experiment.
Experiment SLA[19]	Between *experimenter* and TEFIS; this determines the specific Quality of Service and obligations on both sides for a specific experiment.
Experimenter	The Experimenter is the main user of TEFIS, and one of the three major stakeholders (see Figure 2). It is a legal entity (a person or an organisation) who wants to run an *Experiment* using TEFIS. The Experimenter is the one who owns the *Experiment* and entrusts TEFIS to run it.
Monitoring	Any kind of monitoring – observation of a *Test Run* lifecycle or the low-level instrumentation of a *Resource* for example. Its output is *Test Data*, which is stored in the *TIDS*. From the data management viewpoint, monitoring is simply another activity for which a configuration may be stored and may generate *Test Data* as output.
Resource	A Resource is a building-block component of the service provided by a *Testbed Provider* that enables *Execution Instances* to occur, for example compute, software or storage all are Resources. In the living labs, *Test Subjects* may be regarded as Resources. Resources are described by *Resource Specifications*.
Resource Booking	A description of the type and quantity of *Resource* that is reserved for an *Experimenter* by a *Testbed Provider*. The Resource Booking is usually the outcome of procurement of *Resources*, and may be a *Starting Condition* for an *Execution Instance*, i.e. the *EI* cannot start without the *Resource Booking*.
Resource SLA[19]	Between TEFIS and the *testbed provider*; this determines the resource levels and pricing to be used to run experiments. For example, a reserved timeslot on a particular type of *testbed*. The *experimenter* does not have sight of the *Resource SLA* – TEFIS uses *Resource SLAs* to build up the "complete experiment" service for the *experimenter*.

Class	Description
Resource Specification	A description of a *Resource* that may be used for three purposes: • To describe the *Resource* requirements of an *Experimenter* • To describe the *Resources* offered by a *Testbed Provider* • To describe the actual *Resources* booked by an *Experimenter* at a *Testbed Provider*.
Starting Condition	Starting Conditions are the set of conditions that need to be satisfied before a *Task* or *EI* can begin. For each *Task* or *EI*, the Starting Conditions element is a logical conjunction of *Dependency* specifications of three different types (Task, EI and Data Dependency), and all the specifications included must be satisfied before the *Task* or *EI* can execute.
Task	A Task is a single activity or process from the *Experimenter's* viewpoint – it is a logical process that makes sense (and has value) to the *Experimenter*. A Task is distinct from *Execution Instances* because there may not necessarily be a 1-1 mapping between Tasks and *EI's*. As described in the *Execution Instance* entry, there may be more complex *Execution Instances* hidden inside one Task by TEFIS that the user need not be concerned about, or multiple Tasks may use the same *EI* (see Section 4.2).
TEFIS Portal	This is the portal that provides the front end that Experimenters using TEFIS interact with. Behind the scenes, the Portal communicates with the TEFIS RPRS, which stores the Experiment information in its metadata database. The operator of the TEFIS portal is the second of the three major stakeholders (Figure 2). The RPRS is currently located within the same organisational domain as the TEFIS portal, meaning that the operator of TEFIS provides data management services as well as the experiment management service. This means there is a central data storage and management service, but it can be distributed if required.
Test Data	This is any kind of data stored by TEFIS for the *Experimenter* or in the running of *Experiments*. Test Data may be generated by any TEFIS user, the *Experimenter* or *Testbed Provider* for example, provided they have access rights to store it in TEFIS.

Class	Description
Test Run	A Test Run is a single execution of a *Workflow*. There may be many runs per experiment, for example, different *Workflows* may be evaluated or a parameter space explored (that is separate *Test Runs* using the same *Workflow* but slightly different parameters or circumstances). *Test Runs* in the same test may be executed at the same or multiple *Testbeds*. Because a *Workflow* has multiple *Tasks*, a Test Run may itself have multiple steps (known as *Execution Instances q.v.*), each of which may be run at an individual *Testbed*. Thus an *Experiment* can contain many Test Runs, and a Test Run can contain many *Execution Instances*.
Testbed	The hosting environment for an *Execution Instance*. This may be specific to a certain type of experiment, such as the BOTNIA Living Labs *Testbed* which enables user interaction testing.
Testbed Provider	The organisation providing the *Testbed*. The Testbed Provider is the third and final major stakeholder in TEFIS (Figure 2).
Test Subject	An end-user who interacts with a *Test Run*. The Test Subject is distinct from the *Experimenter*, who owns the *Experiment*: the Test Subject is recruited to exercise the user interface or a dynamic application, for example. A good example of a Test Subject is a recruit in a living lab.
TIDS	The *Testbed Infrastructure Data Service*, a TEFIS specific component providing access to large amounts of test data accessible to authorised users. The TIDS can be installed at the Testbed site and at the Experimenter site. It is the remote storage equivalent to the RPRS *q.v.*
Workflow	A Workflow is a sequence of *Tasks* in a determined order. The purpose of a Workflow is to enable *Task* sequences to be recorded and rerun (possibly with a different configuration) at a later date. A *Test Run* is an instance of a Workflow. See Section 4.4.

The relationship between classes in Table 2 as they relate to the structures and concepts within the Data Services will be described in detail in the following section, as they occur in the three lifecycle stages Experiment Design (Section 6.3.1), Resourcing (Section 6.3.2) and Execution (Section 6.3.3).

6.3. Data Models

In Chapter 5, the three basic stages in the experimental lifecycle were introduced:

- *Experiment Design:* when the experimenter decides what and how to test;
- *Experiment Resourcing (*Resource Procurement): when the experimenter books resource to run the tests designed in the previous stage; and
- *Experiment Execution:* when the experiment test run actually occurs.

In the following sections, each of these stages is described in greater detail in relation to a subset of the domain model for the Data Services. Some classes, as will be seen with *Execution Instance* for example, occur across different stages, providing the conceptual links between each stage as the experimenter moves from design via resourcing to execution. Within each of the model subsections, classes are colour coded:

BLUE	These are *Experiment*, *Task* and *Workflow* components, essentially within *Experiment Design*.
YELLOW	These are *Instance* components (*Execution Instance* and *Test Run* for example) and mainly within *Experiment Execution*.
RED	These are *Resourcing* components, principally from *Experiment Resourcing*.
ORANGE	These are *Dependencies*, and occur mainly at *Experiment Design*.
GREEN	At present, solely the link between *Task* and *Execution*. As stated previously (see Section 4.2), this relationship is beyond the scope of the Data Services.

The detailed meaning of any of the concepts in the models is covered in Sections 4.2ff and Table 2 in Section 6.2.

6.3.1. Experiment Design

This section deals with the design of an experiment – how the experimenter determines the *Tasks* in their experiment, how they are related together (*Dependencies*) and how the *Tasks* may be executed on *Testbeds*. To this end, the concepts of *Workflow*, *Task*, *Execution Instance* and *Dependencies* are gathered together and related to each other in the data model for Experiment Design, shown in Figure 10.

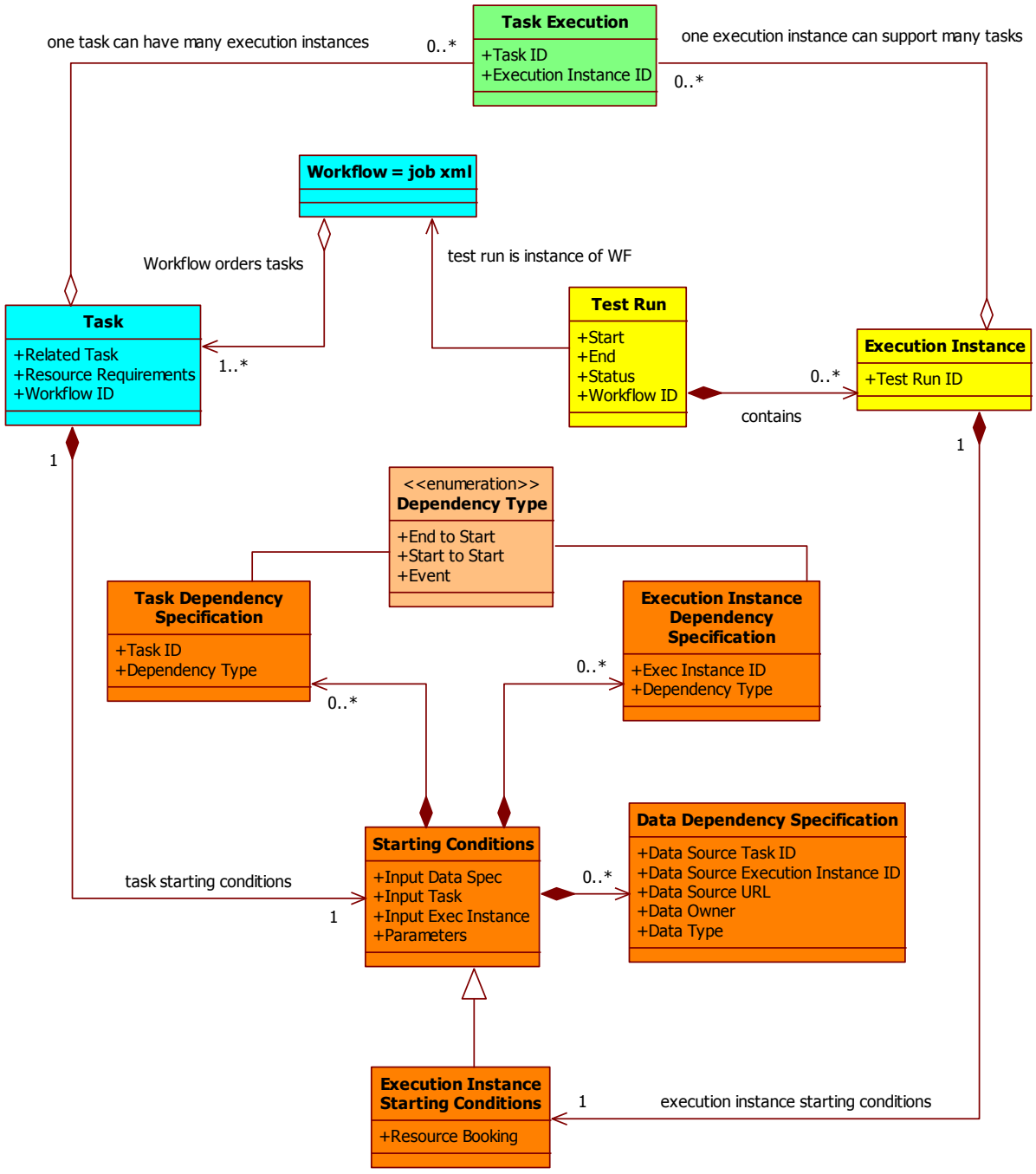

Figure 10: Experiment Design Data Model

As can be seen from Figure 10, a *Workflow*[20] can contain many *Tasks*. This represents the user's creation of the *Workflow* as sequence of *Tasks* that contribute to an overall goal. This is completely user-centric: the view of the *Workflow* and its organisation of the *Tasks*, is specifically aimed at the experimenter and addressing their requirements (Section 4.2). The *Workflow* may also be instantiated into a *Test Run*. The *Test Run* is one run through a *Workflow* (it may be remembered from Section 4.4, a *Workflow* was

[20] Equating *Workflow* and the *job.xml* file simply captures the TEFIS data object – the XML file – used to hold the experimenter's workflow. The technologies that support workflows are discussed elsewhere [3][5].

described as a "program" and therefore a Test Run is running that program once). To accommodate the actual executions on *Testbeds*, the *Test Run* contains *Execution Instances* (EI). A *Task* can also have *Resource Requirements*, which specify what resources are needed for a *Task* (this is covered in more detail later in the *Experiment Resourcing* data model in Section 6.3.2).

Both the *Task* and *EI* can have a set of *Starting Conditions*; this is the mechanism by which the *Dependencies* discussed in Section 4.3 are represented. The idea behind the concept of *Starting Conditions* is that they represent the conditions that need to be satisfied before a *Task* or *EI* can begin. For each *Task* or *EI*, the *Starting Conditions* element is a logical conjunction of specifications of three different types (*Task, EI* and *Data Dependency*), and all the specifications included in the *Task* or *EI* must be satisfied before the *Task* or *EI* can execute. For example, we can specify that an *EI* depends on a *Task* completing, an *EI* producing some data, and a second *Task* signalling an event. All these conditions must be satisfied before our *EI* can run, and the *Starting Conditions* element for our *EI* can hold the specifications for all the conditions required that enable the *EI* to execute.

A special case of *Starting Conditions* is that for an *EI*. In addition to the attributes required for a *Task*, an *EI* also requires a resource on a *Testbed* in order to be executable. Therefore an additional subclass of *Starting Conditions* for *Execution Instances* is included, with a reference to a *Resource Booking*. *Resource Bookings* are discussed later (Section 6.3.2, Figure 11); briefly they represent the reservation of a certain amount of resource on a *Testbed* to support *EI*'s.

6.3.2. Experiment Resourcing

This section deals with the resourcing stage of the experiment lifecycle. Resourcing is required so that the Experimenter can use testbed resource to run their Experiment. Figure 11 shows the part of the data model pertaining to resourcing. Some elements from Figure 10 are shown here, indicating where there are joins and overlaps between the Experiment Design and Resourcing.

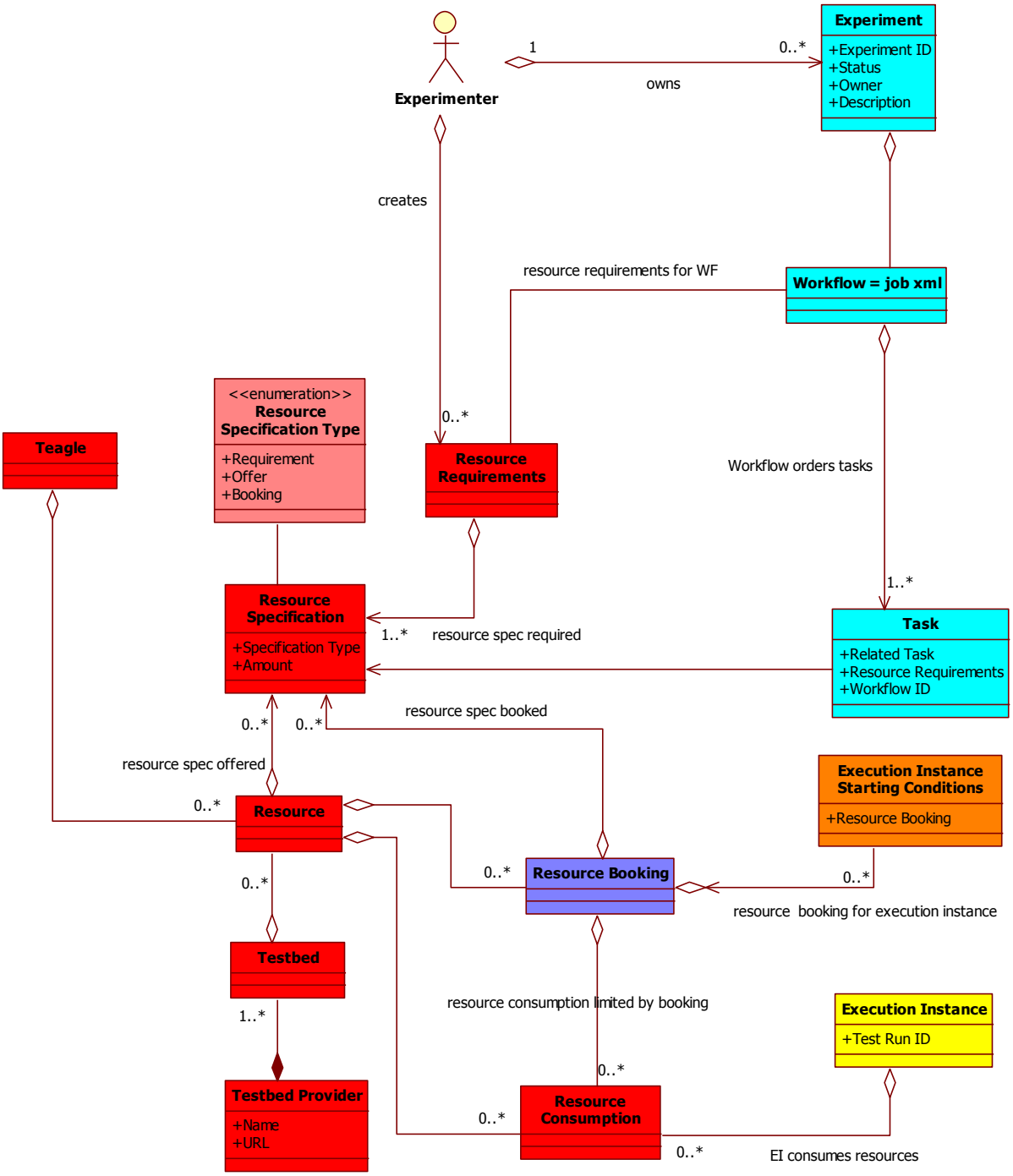

Figure 11: Experiment Resourcing Data Model

As with Figure 10, logical groups are identified by colour. For the task and workflow components (shown in blue), *Experiment, Task* and *Workflow* are carried over from the previous *Experiment Design* model to indicate the relationship between the *Tasks* in the *Workflow* and the *Resource Specification:* a *Workflow* depends on certain resources being available (the *Resource Requirements*), but it is the individual *Tasks* which determine the type and configuration of the resource to be used (in the *Resource Specification*). The *dependency – Execution Instance Staring Conditions –* also relates back to *Experiment Design*, showing how the reservation of resource of a given specification is a necessary assumption.

In Figure 11, the Experimenter owns an *Experiment*, which has a *Workflow*. *Workflows* have been already discussed in connection with the model in Section 6.3.1, as well as in some detail in Section 4.4; the *Experiment* entity is a container for a *Workflow*. It is possible to have more than one *Workflow* per *Experiment*; the decision about how many *Workflow*s there are per *Experiment* is up to the Experimenter. Recalling the Experiment Design data model (Figure 10), each *Workflow* is a sequence of *Tasks*, and each *Task* has its own *Resource Requirements*. Because of this containment relationship, by looking at the *Workflow*, the experimenter can determine the resources needed for their *Workflow*, and these are recorded in the *Resource Requirements* element. Recalling the resource procurement cycle in Section 4.5 above, the *Resource Requirements* element acts as a "shopping list" for the Experimenter in terms of the resources required for their *Workflow*. The *Resource Requirements* also map onto part of the VCT (Virtual Customer Testbed) model (described in [4]), the part describing the user's requirements.

On the *Testbed Provider's* side, the *Testbed Provider* owns *Testbed*s, each of which provide *Resources*, and the *Resources* made available by the *Testbed Providers* are recorded in Teagle [4].

The *Resource Specification* element is used in three different ways corresponding to the different descriptions of resources described above in Section 4.5 on resourcing. Firstly it describes the resource requirements of the experimenter, secondly it describes the resources on offer at a Testbed, and finally it describes the resources booked by an experimenter for an experiment. These specification types are described by the *Resource Specification Type* enumeration. The specification type cannot be mixed – the same *Resource Specification* object cannot be both a specification of the Experimenter's requirements and also describe the resources on offer. Not only are these types mutually exclusive due to their enumeration types, the specifications have different owners (in the example the Experimenter and the Testbed owner), and different purposes. Different objects should be created to represent each *Resource Specification*, so the Experimenter can use one *Resource Specification* object to represent their requirements and the *Testbed provider* can use another *Resource Specification* object to describe what is on offer.

As described in Section 4.5, the outcome of the resource procurement stage is a *Resource Booking*, and this is included in the data model, pointing at the *Resources* and the *Resource Specification* objects. The booking of resources is recorded separately from the *Resource Specification*, as the actual *Resource Booking* may refer to an instance that may need to be explicitly identified (for example we might need to know which software licence number is reserved for an experiment in addition to the specification of the software). A *Resource Booking* is also a type of *Starting Condition* for an *EI*, so there is a link between the *Resource Booking* and the *Execution Instance Starting Conditions*.

6.3.3. Experiment Execution

The final stage of the experimental lifecycle deals with the execution of the Experiment. This section is more concerned with metadata for physical items than the other two data models, since when an experiment is running, the Experimenter and other users will need to store and retrieve data using TEFIS. There are thus many links between this section and the Physical data model described in Chapter 7.

The actual data model for Experiment execution is shown in Figure 12. As with the previous data models, the Workflow and Task elements are shown to indicate cross over with other fragments shown in Figure 10 and Figure 11. The colour coding is the same as in those figures.

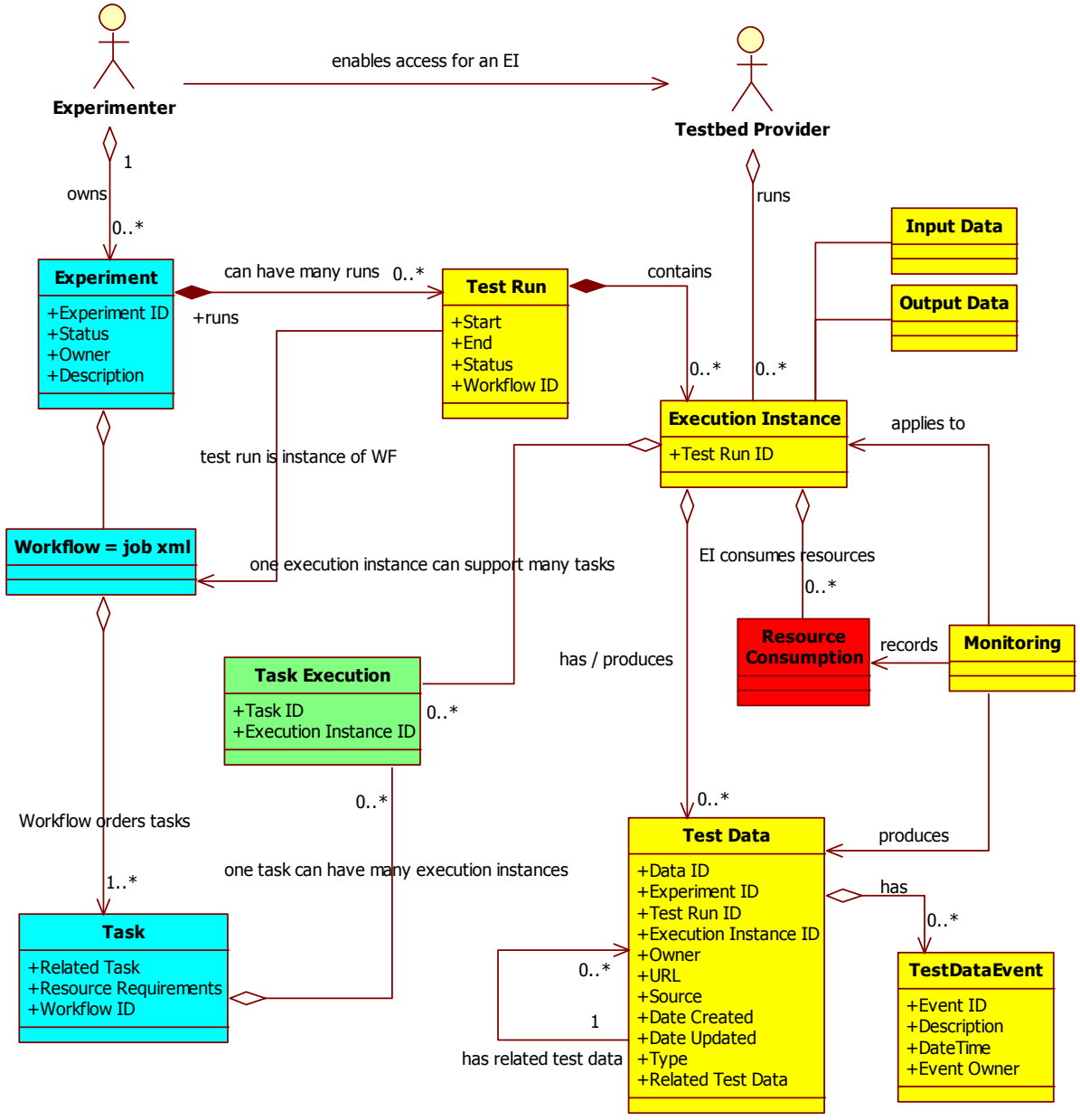

Figure 12: Experiment Execution Data Model

The central elements in this data model are the *Execution Instance* and *Test Data*. The *Execution Instance* (*EI*) has already been discussed, and represents a single execution at a *Testbed*. *Input* and *Output Data* are aspects of the *EI*, and are intended to identify clearly what is regarded as input (which may be read-only for the *EI*) and output (which the *EI* can write to). These are implemented as subfolders of an *EI* in the physical model (Chapter 7 below).

Test Data represents any data that is required or generated during the execution on a *Testbed*. The *Test Data* entity is intended to provide basic metadata information for any data item, and any item stored in the data management will have at least some of the attributes in the *Test Data* element applied as metadata. For example in the iRODS storage system implemented in WP6, any file uploaded

automatically has the experiment ID applied to it as metadata. In addition, because the iRODS storage system is REST-based, each data item will have its own unique URL to address it.

The *Test Data Event* is used to record any type of event that occurs on *Test Data* items. This is a mechanism to address provenance requirements (see Chapter 5 and Table 3, in 11: Appendix II), and provides a simple way to record the actions that occur on *Test Data* items. Every time data is added, removed or changed, an event should be recorded describing the nature of the event, when it occurred and who or what instigated it.

Finally, the *Monitoring* element is a link to the monitoring work done elsewhere in the project [2]. We assert that *Monitoring* applies to an *EI*, because the *EI* is what is executed at the Testbed and is therefore monitored. We also assert that *Monitoring* produces *Test Data*. From the viewpoint of data management, the output of *Monitoring* can be regarded as *Test Data* like any other. Monitoring data may have specific metadata to identify it as monitoring output or configuration, but it should require no special treatment.

One particular use of Monitoring is the recording of *Resource Consumption* and *Performance* information. This has already been discussed in connection with the Resourcing data model in Section 6.3.2, but *Monitoring* of the running of the *Execution Instance* is the source of the *Resource Consumption* information.

Finally, there is a link between *Task* and *EI*. This represents the different relationships a *Task* can have with an *EI*, as discussed in Section 4.2. The link is named "*Task Execution*" and is a particular instance of a Task running on an EI. In the physical model, this is implemented as a folder beneath a *Test Run*, so any data that is the result of *Task Execution* has a storage location.

7. Physical Storage Models

The TEFIS portal hosts a data management component that is used as a centralised store for experiment data. Experimenters can create experiment folders and allow testbeds access to the data within these folders to run tests and to upload output data, which the experimenter can then see. To organise the data in the TEFIS portal, we require a physical data model, and there are a number of requirements for this storage[21]:

- Different experimenters' data needs to be segregated from each other for security;
- If an experimenter has multiple experiments, these should be segregated from each other to avoid confusion;
- Different actors will need to access different subsets of experiment data. For example, when different testbeds are involved in the same experiment, they need to be given access to the input data for their part of the experiment, and to be able to write their output to a particular location without fear of another testbed overwriting their data;

[21] These requirements are implemented as refinements to Requirements 4, 7, 13 and 19 in 11: Appendix II, Table 3

- The experimenter should be able to enable / disable access for different testbed providers to subsets of their experiment data[22].

The basic philosophy behind the physical data model is to provide *just enough* structure so as to satisfy the requirements above - we do not want to be over-prescriptive to experimenters and testbed providers. Within the structure we have provided, experimenters and testbed providers should be free to organise the data any way they see fit. Specific folders within the structures for a given experiment are created exclusively for use by the Experiment Manager in support of the individual TEFIS experimenter and the experiments and testruns they create[22].

The physical storage model is implemented as a folder structure in the iRODS server at the TEFIS portal. It should be noted that some folders, as marked in ***bold italics*** in Figure 13, are reserved they are generated by TEFIS in support of experiment activities and are need to support the successful operation of a given experiment and the associated testruns. In addition, some of the reserved folders are exclusive to the Experiment Manager to store important objects in support of this specific experiment and their experiment.

The folder structure is derived from the organisation of parts of the metadata models that are appropriate for experiment execution (see Figure 12) and provides a storage facility with some organisational capability for *Experiments*, *Test Runs*, *Tasks* and *Execution Instances*. Here these concepts are implemented as folders. In addition metadata can be added to any of the data items such as files and folders. iRODS has a useful mechanism that permits metadata in the form of attribute-value pairs to be applied to files and folders (in some cases automatically), so the concepts and attributes from the metadata models can be applied to folders and files within the folders.

An additional requirement guiding the design of the physical storage model is the need to provide access control for different users[23]. This is beyond the scope for this deliverable, which concerns the metadata models. Briefly, there is a need to prevent unauthorised users reading and writing to experimenters' data, but to allow authorised users to access what they need to. For example, an experiment should be accessible only to its owner, unless the owner grants access to other users (a significant case in point is a testbed provider so they can use experiment data to set up a test run). It is considerations such as these that have also guided the design of the folder structure[24], which is summarised in Figure 13.

[22] It should be noted that each time a user account is created for the Data Services, a corresponding Experiment Manager account will be created, specific to this user and to be used by the Experiment Manager for objects created during the experimental lifecycle. In some cases, the individual user may be able to view Experiment Manager generated objects and files, but they will remain protected and may not be modified by the user to ensure successful operation of the experiment and associated testruns.

[23] See Requirement 13, Table 3, 11: Appendix II

[24] iRODS access control can be used to enforce access rights: there is an IRODS user account for each experimenter and each testbed provider (plus some other dedicated accounts such as the administrator and experiments manager). Using iRODS user accounts to represent the actual stakeholders in TEFIS means we can utilise iRODS' built-in access control for files and folders within the experiment. An iRODS user automatically owns any folder or file created within their home area, and can enable / disable read and write access to other users.

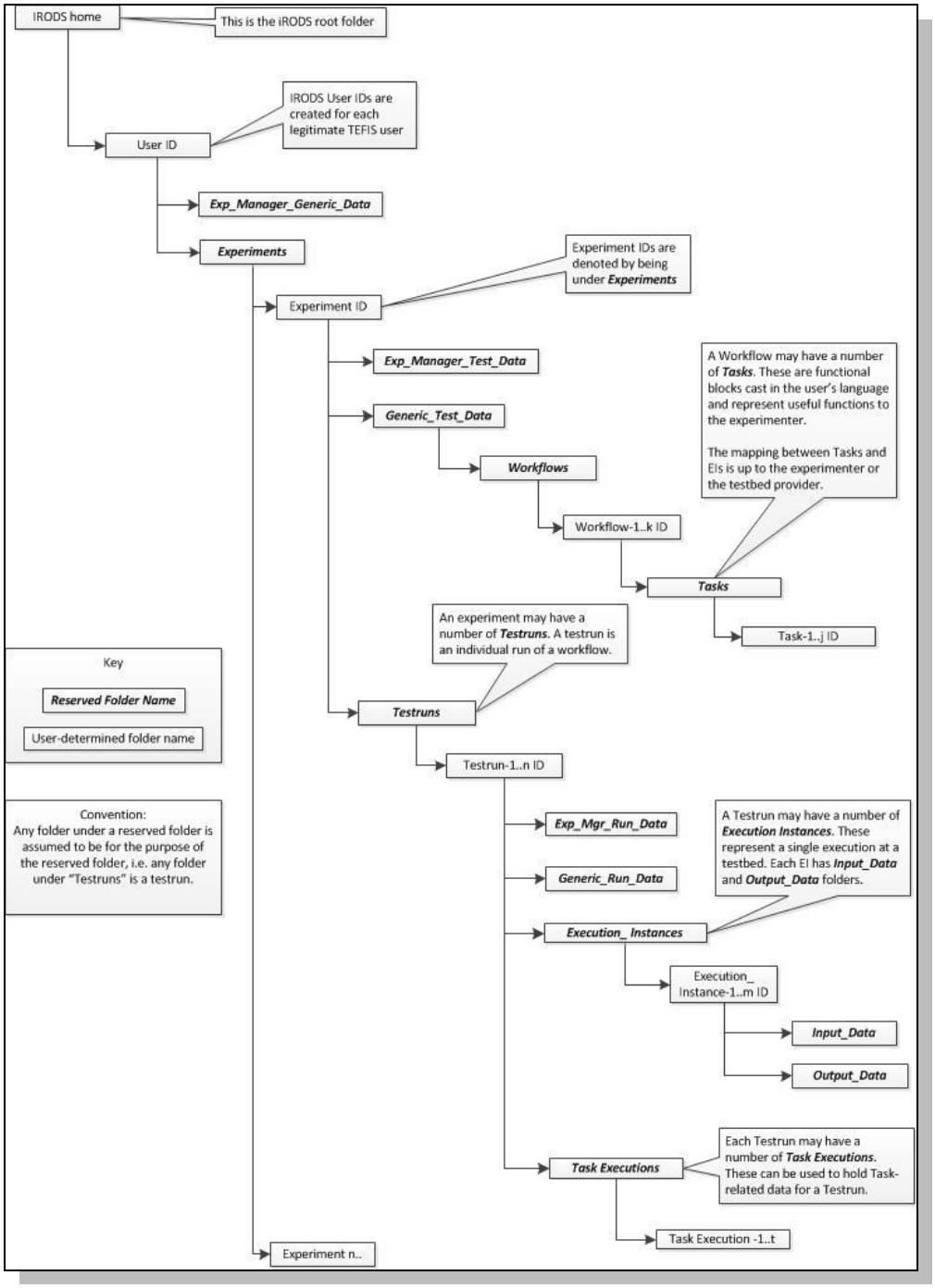

Figure 13: Data Services Folder Structure

Using this structure, where each user's experiments are stored beneath their home folder means that the user is the owner of the experiment, and can utilise iRODS' access control to manage access to other users (notably the testbed providers can be given access to the process folders).

The iRODS home is the access point for the Data Services in the TEFIS portal. Every time a legitimate user registers, an iRODS user account is created for them with a home folder (this is "user ID" in the figure). All data for that user is stored within this home folder. The user is free to store anything they like in their home folder (subject to acceptable usage, terms and conditions and charging policies), but if they want to store an experiment, they must use a pre-defined folder structure, which is shown in the remainder of Figure 13. This does *not* mean that all objects within the folder structure will be visible and therefore used by the Experiment Manager or any other TEFIS components. The TEFIS experimenter must create experiments to be run via TEFIS using the defined process to ensure that the necessary objects are generated and available at execution time.

A REST method is supplied (one of a number, described in [5]) to let the user create an experiment. This method creates the folder structure beneath the "Experiments" folder. The user supplies the experiment ID, and this is used as the top folder for the experiment. Under this, a "Generic Test Data" folder is created, where the user can store anything that is not test run, Execution Instance, or Task specific.

For each workflow in the experiment, the user can request a place to hold their data and Tasks. This is done using a REST method that creates a "Workflow" folder beneath the "Generic Test Data" folder. The method takes an argument that indicates the number of tasks in the workflow, and under the workflow folder, a folder is created to hold data for each task in the workflow.

When an experiment is created, the user can create a Test run with another REST method. This creates the folder structure beneath "Testrun-1..n". The Testrun folder corresponds to the Test Run element in Figure 10 and Figure 12, and is the main holding place for all data associated with a single test run. As described previously, the test run is an instance of a Workflow representing one run through of the complete workflow. This means that each (user focused) *Task* and each (testbed and application focused) *Execution Instance* needs a storage place for data generated throughout the test run. To represent the tasks executed in the test run, a set of "*Task Execution*" folders are also created beneath the test run folder. These can be used to hold any data that pertains to the tasks executed in the test run. In Figure 12, this is represented by the green element, "*Task Execution*", which joins the *Task*s in the *Workflow* to the EI's being executed on the testbeds. Similarly, a set of "*Execution Instance*" folders are also created beneath the test run folder to hold data pertaining to the actual executions on testbeds. As described previously, the mapping between the *Tasks* and *EI*'s is outside the scope of data management (see Section 4.2), so the REST method to create a test run simply takes arguments to indicate how many tasks and EI's a test run has, and creates the appropriate folders accordingly. There are also methods to link *Tasks* and *EI*'s through the metadata – once a user knows the mapping between *Tasks* and *EI*'s, they can use the methods to update the metadata on the *Tasks* and *EI*'s.

The folder structure represented in Figure 13 has been through a number of iterations (see [6] and [7]) as experience with the initial TEFIS use cases has been taken into account for the implementation of the Data Services. The overall approach, however, remains very similar: the path within the folder structure is used for the rapid identification of the location of in- and output data associated with a specific test run; metadata associated with the folders and any files within the folders is used to identify ownership, provenance and links between objects.

8. Conclusion

This document discusses modifications to the *Specifications of Experimental Metadata* previously outlined ([1]) and in light of the experience gained from the initial implementation and integration work during the first period of the TEFIS project. At that time of the original deliverable, a number of assumptions were made prior to the definitive selection of a suitable supporting technology, as well as direct engagement with the first of the TEFIS test scenarios. As a result of both of those, and integrating the technology with other TEFIS components, a number of modifications were deemed appropriate.

To some degree, in looking more closely at the initial TEFIS test cases, we have moved from the TEFIS-*internal* focus outlined in the first version of this deliverable to a more TEFIS-*external* approach (see Conclusion, [1]). Modifications in this deliverable based on the initial analysis of the first three TEFIS test cases (Chapter 4) has led to a simplified Experimental lifecycle (Chapter 5) as well as a different view on the underlying data model (Sections 6.2 and 6.3, and the related sub-sections), capitalising on what is appropriate in related metadata standards (Section 6.1). In turn, this also had implications for the physical data storage structures (Chapter 7), which needed to provide a logical architecture for the TEFIS internal components, the TEFIS testbed providers and experimenters using the TEFIS platform to run their experiments.

With these modifications and adjustments in place, we are now in a better position to move forward with the benefit of experience, in support of the TEFIS Open Call experimenters as well as new testbed providers and the broader FIRE community, whilst maintaining sufficient continuity with the initial internal data management view. The TEFIS Data Services are now in a position to support all three of our main stakeholders: TEFIS operators (and internal components), TEFIS testbed providers, and most importantly, the TEFIS experimenter community.

9. References

[1] TEFIS Deliverable D6.1.1 (2010) *Specifications for Experimental Metadata*

[2] TEFIS Deliverable D2.1.2 (2011) *Global architecture and overall design*

[3] TEFIS Deliverable D4.1.1 (2010) *Test toolkit implementation*

[4] TEFIS Deliverable D3.1.1 (2010) *Teagle assessment and TEFIS portal specifications*

[5] TEFIS Deliverable D3.2.1 (2011) *Building blocks integrated into TEFIS Portal*

[6] TEFIS Deliverable D6.2.1 (2011) *RPRS Prototype*

[7] TEFIS Deliverable D6.3.1 (2011) *TIDS Prototype*

[8] Brand, A., Daly, F. and Meyers, B. (2003) *Metadata Demystified*, The Sheridan Press/ NISO Press, ISBN: 1-880124-59-9 http://www.niso.org/standards/resources/Metadata_Demystified.pdf

[9] NISO Press (2004) *Understanding Metadata,* National Information Standards Organization, ISBN: 1-880124-62-9 *available at* http://www.niso.org/publications/press/UnderstandingMetadata.pdf

[10]Matthews, B., Sufi, S., Flannery, D., Lerusse, L., Griffin, T., Gleaves, M. and Kleese, K. (2010) *Using a Core Scientific Metadata Model in Large-Scale Facilities,* The International Journal of Digital Curation (Issue 1, Vol 5), *available at* http://www.ijdc.net/index.php/ijdc/article/viewFile/149/211

[11]Riede, M., Schueppel, R., Sylvester-Hvid, K.O., Kühne, M., Röttger, M.C., Zimmermann, K. and Liehr, A.W. (2010) *On the Communication of Scientific Results: The Full-Metadata Format,* Comput. Phys. Commun. (181:651-662), *available at* http://arxiv.org/abs/0904.1299

[12] Jones, M.B., Berkley, C., Bojilova, J. and Schildhauer, M. (2001) *Managing Scientific Metadata*, IEEE Internet Computing (September – October, 59:68), *available at* http://knb.ecoinformatics.org/distributed-data-ic-final.pdf

10. Appendix I: *Metadata Schema*

The metadata schema below corresponds to the experiment folder structure shown previously. This is not strictly enforced by iRODS or associated tools, but provides the necessary structure for data being submitted to the RPRS and TIDS. The attributes defined form part of an attribute-value pair, which describes some aspect of a data item or folder.

The numbered headings indicate an inheritance hierarchy with subsections inheriting from the parent with the same index number (1.1 from section 1; 1.2.1 from section 1.2; and so on). The schema may be updated on request to satisfy data requirements from other components or external testbeds as agreed by the TEFIS Data Services.

```
1. Experiment Info // related to Dublin Core
Experiment-Info.Data-ID
Experiment-Info.Experiment-ID
Experiment-Info.Testrun-ID
Experiment-Info.Execution-Instance-ID // replaces Process-ID
Experiment-Info.Owner
Experiment-Info.Source
Experiment-Info.Date-Created
Experiment-Info.Date-Updated
Experiment-Info.Type
Experiment-Info.Related-Experiment-Data // related to CSMD
Experiment-Info.URL


    1.1 Input Data // Provenance (& Curation)

        1.1.1 Application

        Application.Name
        Application.Version
        Application.Source-Location
        Application.Static-Parameters

        1.1.2 Application Config

        Application-Config.App-ID
        Application-Config.Config-Parameters

        1.1.3 Testbed Config

        Testbed-Config.Platform
        Testbed-Config.Settings
        Testbed-Config.Requirements
```

Testbed-Config.Output-Data-Location
Testbed-Config.Operating-System
Testbed-Config.Protocol
Testbed-Config.Device

1.2 Output Data

Output-Data.Testbed-Provider-ID

1.2.1 Resource Usage Data

Resource-Usage-Data.CPU-Usage
Resource-Usage-Data.Disk-Usage
Resource-Usage-Data.Network-Usage
Resource-Usage-Data.Memory-Usage

1.2.2 Context

Context.Operating-System
Context.System-Memory
Context.Available-Memory
Context.Processor-Architecture
Context.Virtual-Machine-Image
Context.Total-Machine-Load

1.2.3 Testbed Provider

Testbed-Provider.Name
Testbed-Provider.URL

1.2.4 Testbed Resource Monitoring

1.2.5 Application Test Output

2. Experiment Profile **// related to Dublin Core**

Experiment-Profile.Experiment-ID
Experiment-Profile.Status
Experiment-Profile.Owner
Experiment-Profile.Description
Experiment-Profile.Keywords
Experiment-Profile.Related-Work **// related to CSMD**
Experiment-Profile.Derived-From **// related to CSMD**

3. User **// related to Dublin Core**

```
User.User-ID
User.User-Name
User.User-Email-Address
User.User-Affiliation
User.Contributor
User.Contributor-User-ID      // relates back to User record for
this contributor

4. iRODS Metadata tags

These are tags that can be used to denote something to the iRODS'
query processor, for example "Collection.Type" is used to denote
a particular type of a collection, for example "User" or
"Experiment".

Collection.Type
 = {User, Experiment, Testrun, Process, Process-Input-Data,
Process-Output-Data}
```

11.Appendix II: *Technology Choice*

Both for the RPRS and TIDS, we have selected the data management system iRODS[25]. This selection was made in light of the following requirements:

Table 3: Extract[26] from Table 9 ([1])

Summary of requirements for data management and data services in TEFIS

No.	*Requirement*	*Description*	*Data affected*
4	Record test-related data	TEFIS must be able to receive and convert, if necessary, information relating to the test and its setup.	Configuration data, application data (including application and/or workflow).
7	Provide access to test data	TEFIS should enable data to be viewed by the owner and any other authorised user.	All data.
8	Allow remote data transfer	TEFIS should be able to manage the transfer of remote data from one location to another, even though this may not involve transfer via the TEFIS platform itself. This may result in changes to the references in the test profile. (See requirement 21, which describes the same requirement from a data management perspective.)	Experimental data.
12	Record information about any conditions in the environment when a test-run was executed.	In addition, any specific experimental context information will need to be written to the test profile.	Context, including static information, such as platform characteristics, as well as dynamic information such as any other tests or applications running at the same time.

[25] https://www.irods.org/index.php/IRODS:Data_Grids,_Digital_Libraries,_Persistent_Archives,_and_Real-time_Data_Systems

[26] Requirements in the original table which relate more to data management processes and which are handled in association with other TEFIS components have been removed from this extract for clarity.

No.	*Requirement*	*Description*	*Data affected*
13	Enforce access rights	TEFIS should validate a requester's right to access and interact with any information within the platform and managed by the platform (such as data helped at remote test-bed facilities and accessed via TIDS).	All data, local (ie. Stored within the RPRS) or remote (ie. Accessed via TIDS).
14	Record test-relations	TEFIS needs to be able to cross-reference any data, such as configuration, results and runtime context, which relate to a specific test or test-run.	All data types.
15	Record test management information	TEFIS needs to be able to handle all data which identifies a test.	Owner, type and where run (ie., test-bed ID or federated resource descriptor).
16	Generate test result reports	TEFIS must be able to provide links or references to raw experimental data output (held on the test-bed platform and accessed via TIDS) so that suitable summary reports can be created.	Experimental data.
17	Record test-run management information	TEFIS needs to update any test profile with appropriate identifying information about individual test-runs.	RunID, parent TestID; date, time; link to remote data (raw or consolidated data).
19	Record experimental data, that is data produced as output from a given test-run	TEFIS needs to be able to retain links or references to the data generated during a given test-run.	Experimental data.
20	Record provenance	TEFIS should retain sufficient information within the test profile to be able to identify data or object history and source(s).	All data.
21	Enable data transfer	TEFIS should be able to allow a user to retrieve or transfer remote data from one or multiple sites to be transferred externally to TEFIS to their own	Experimental data.

No.	*Requirement*	*Description*	*Data affected*
		server location.(See requirement 8 which describes this from a data retrieval perspective.)	

In light of these requirements, we evaluated a number of open source, off-the-shelf systems to find one or a combination that addressed these requirements or would allow us with suitable extension and integrated into the TEFIS infrastructure to address them. Initially, we investigated asset management systems (two evaluated were Fedora[27] commons and DSpace[28]), which provided user interfaces, metadata markup and search features. However these alone did not provide native storage of experimental data files, and so lower-level storage approaches were investigated. The early front runner (and eventual winner) was iRODS[25].

A key attraction with iRODS for TEFIS is that it is mature and well-adopted. iRODS is derived from SRB, which has about 10 years' development and is used in many applications across the world. It has a lively community around it, and all this means that its future is much more certain than less well-adopted systems. There are many extensions and the latest release (2.5) appeared in February 2011. The core collaborators[29] include:

- CCIN2P3, the national high-performance computing centre in Lyon
- The United Kingdom eScience program
- SHAMAN, Sustaining Heritage Access through Multivalent ArchiviNg
- The University of Liverpool
- ARCS, the Australian Research Collaboration Service
- CeRch, the Centre for e-Research at King's College, London
- KEK, the High Energy Accelerator Research Organization, Japan

and is installed[30] across US federal agencies, as well as worldwide supporting the data needs across the physical and natural sciences, digital archiving, geosciences, high-performance and grid computing, engineering, education and industry.

We originally intended to partner iRODS with one or other of the asset management systems – iRODS would provide the low-level data storage and the asset management systems would provide the metadata management and searching and so forth: iRODS provides support for integration with Fedora Commons and DSpace. However, as iRODS was further evaluated and as we began to provide access to it

[27] http://fedora-commons.org/
[28] http://www.dspace.org/
[29] See https://www.irods.org/index.php/Collaborators
[30] See http://diceresearch.org/DICE_Site/iRODS_Uses.html

to tour TEFIS partners, it was found that it provides many of the asset management features alone. Its key technical features include:

- Distributed or central file storage – one to many iRODS servers can be used, all connected to a central data catalogue (see Requirements 8, 16, 19, 21 in Table 3);
- iRODS folders and files can have user-defined metadata applied to them. This can be searched so folders and files can be found (see Requirements 4, 12, 14, 15, 17, 19, 20 in Table 3);
- iRODS provides user identity management and folder and file permissions associated with user IDs (in a similar way to a unix file system) (see Requirements 7, 13, 20 in Table 3);
- iRODS provides automation in that automatic "rules" can be created that can be triggered on certain conditions and which execute certain processing (this supports additional TEFIS-internal requirements, and integration with other components).

iRODS addressed the major requirements of TEFIS, and because of this and its maturity and level of adoption, it was chosen as the underlying technology to base the data management component on.