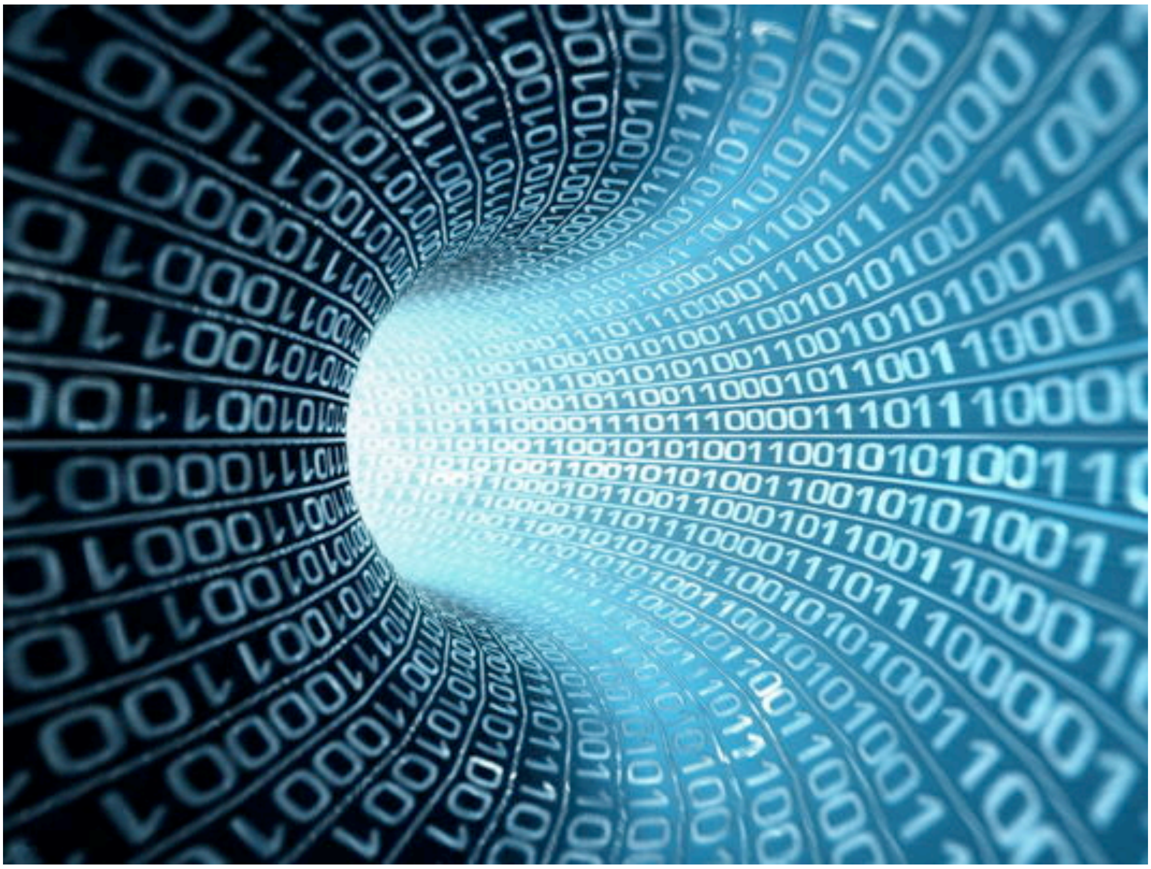
Preparing for Data-driven Infrastructure

JISC Observatory TechWatch Report
Final report v1.0 September 2012

Author: Max Hammond

Produced by JISC Observatory • http://observatory.jisc.ac.uk/



JISC Observatory
Observing trends in innovation

JISC cetis

UKOLN

About this report

Managing data is a strategic problem for HE institutional managers and a technical problem for IT staff. This report provides an overview of some concepts and pragmatic approaches as well as tools that can help address this strategic and technical problem.

Specifically, this report: 1) describes data-centric architectures; 2) gives some examples of how organisations are already sharing data and discusses this from a data-centric perspective; 3) introduces some tools and technologies that can support data-centric architectures as well as some new models of data management; 4) concludes with a look at the direction of travel. This report also provides a glossary to help clarify key terms and a 'References' section listing works cited.

This report is informed by feedback gathered during an open commenting period from 29 June through 16 July 2012. The final version of this report has been made available in the TechWatch section of the JISC Observatory Website.

HOW TO CITE THIS REPORT

Max Hammond, *Preparing for Data-driven Infrastructure*, September 2012 [Online] Available at: http://blog.observatory.jisc.ac.uk/techwatch-reports/data-driven-infrastructure/

ABOUT THE AUTHOR

Dr Max Hammond is a consultant who has worked widely across the education and research sectors, focusing on how technology supports business at a strategic level. He has extensive experience of how new technologies are adopted by institutions as well as the wider perspective of developing Internet information systems. More information is available from his Website < http://www.mh-strategy.eu >.

ABOUT CC-BY LICENSING OF THIS REPORT

Foreword

Higher Education institutions face requirements for greater transparency in their business processes, with strong demands for better access to higher-quality information. The Key Information Sets (KIS) now required by HEFCE are one example of this. The detailed reporting required in the Research Excellence Framework (REF) is another. Furthermore, as our institutions deal with a significant number of sector-wide bodies (such as HESA, UCAS, QAA), each of these agencies is demanding more robust data management and the sharing of good-quality information.

Through the Interim Regulatory Partnership Group (formed in response to the changing regulatory framework for Higher Education in England), HESA commissioned a far-reaching information landscape project.[1] In its report submitted on 15 June 2012, this project recommended that the sector as a whole should collaborate to model, standardise, and facilitate the reliable exchange of high-quality management information. The report also noted that some HE institutions "struggle to provide timely and accurate data".[2]

As our HE institutions face increasing requirements to manage data more effectively, this could mean, for some institutions, a shift in emphasis of systems design towards a 'data-centric architecture'. In any case, if our institutions are to exploit an emerging data-driven infrastructure, they will need to understand what this entails. This report, the third of the JISC Observatory's TechWatch series, is a very good place to start.

In parallel with this HESA project work, management of research data is also being established as a *bona fide* research output. Requirements from funders are gradually being introduced, mandating that research data must be properly

[1] For a general overview of HESA work on improving and consolidating data flows across HE institutions, see: http://www.guardian.co.uk/higher-education-network/blog/2012/feb/13/university-speaking-our-data-language. See also the HESA Website for more information about its 'Redesigning the higher education data and information landscape' project: http://landscape.hesa.ac.uk/about-the-project/.

[2] See page 3 of the HESA report's Executive Summary: http://landscape.hesa.ac.uk/wp-content/uploads/2012/01/IRPG_PrjB_Final_Report_Exec_Summary.pdf .

managed and made available.[3] As research, especially in certain disciplines, is becoming more data-intensive, expectations of increased access to research datasets are growing.[4]

Within this context of increasing regulation from government and changing requirements from HE agencies and other stakeholders, our institutions need to find a sustainable approach to managing data.

Evolving approaches to data management

Approaches are evolving to the development and deployment of information services both within and outside our teaching and research institutions. It has been accepted for some time that closely coupled, vertically integrated, or 'monolithic' applications can lead to what are called 'data silos' where information becomes locked into particular systems and inaccessible to other processes and applications. In response to this, some institutions and many systems vendors have advocated a 'service-oriented-architecture' (SOA), one which is designed to break down silos into modular components of functionality.

However, on the Web we have seen the rise of relatively simple technical approaches to making data available through so-called 'RESTful APIs', and the data 'mashup' is now a staple of many successful Web 2.0 applications. Behind this paradigm is the notion of a 'resource-oriented architecture' (ROA), and there is growing interest in the benefits that this approach might offer organisations, especially those with a desire or a requirement to make data openly available. It is, perhaps, a natural step from 'resource-oriented architecture' to 'data-oriented architecture'.

An alternative design philosophy has started to gain momentum in the last few years, one that puts data at the heart of the organisation and its infrastructure. Tools, technologies and techniques are emerging to help organisations examine, model and manage their data assets in a holistic and coherent way. This is, in part, being driven by the broad and growing interest in 'analytics' and business intelligence within our institutions as well as from central bodies with a sector-wide remit.

In addition to these drivers, alternatives to the established 'relational database' paradigm offer new approaches to the management of data, especially in situations where the datasets in

[3] See introduction to the Digital Curation Centre's Data Management Planning Tool at https://dmponline.dcc.ac.uk: "Funding bodies increasingly require their grant-holders to produce and maintain Data Management Plans (DMPs), both at the bid-preparation stage and after funding has been secured."

[4] See http://www.ariadne.ac.uk/issue65/wilson-et-al for an overview of work on 'Developing Infrastructure for Research Data Management at the University of Oxford' as well as broader discussion of the increasing importance of managing research datasets elsewhere.

question are on a very large scale, or where diverse and dynamic datasets are best handled in a less restricted or 'normalised' manner.

Understanding and exploiting data-driven infrastructure

To meet the new regulatory requirements in an efficient and sustainable way, the HE sector has seen the emergence of innovative approaches to 'data-driven infrastructure' where it is access to data (from institutions and agencies) that determines the shape and function of that infrastructure.

As our HE institutions face increasing requirements to manage data more effectively, this could mean, for some institutions, a shift in emphasis of systems design towards a 'data-centric architecture'. In any case, if our institutions are to exploit an emerging data-driven infrastructure, they will need to understand what this entails. This report, the third of the JISC Observatory's TechWatch series, is a very good place to start.

Paul Walk, Innovation Support Centre at UKOLN, August 2012

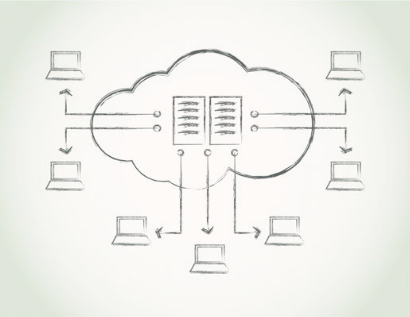
Executive summary

All organisations are required to share data in some ways – whether internally, to a defined community, or openly. Data sharing is often held to be a benefit in its own right, in particular within the education and research communities, but there are strong and increasing drivers to share more, and more openly.

Institutions have adopted a wide variety of approaches to creating data architectures internally. In many cases, these approaches are developed *ad hoc*, with data flows created between systems as needed, and may rely on tightly coupled interfaces, or even manual processing. A few institutions have begun publishing open data, and are using this approach to drive internal systems.

A fully fledged data-centric architecture requires a level of technical maturity that may be challenging for some organisations and could be expensive and high-risk to implement quickly. There are, however, pragmatic steps that HE and FE institutions can take now to begin moving toward a more data-centric approach, and these steps are the focus of this report.

One concept in systems design that is growing increasingly important to support these factors is what is termed data-centric architecture. This approach to building and buying systems focuses primarily on the organisation's data (as opposed to its systems) and is intended to facilitate sharing data between the processes that need it. Implementing this kind of architecture requires a fundamental reconsideration of the relationships between systems.



A fully fledged data-centric architecture requires a level of technical maturity that may be challenging for some organisations and could be expensive and high-risk to implement quickly. There are, however, pragmatic steps that HE and FE institutions can take now to begin moving toward a more data-centric approach, and these steps are the focus of this report. Many institutions could consider developing appropriate APIs to their systems as a sound first step in this direction.

As with any development activity, it is fundamentally important, when establishing where and how to create APIs, to consider who the end-users will be, who will be creating applications using your API, what use cases you expect to meet, and in which usage scenarios. APIs should not be created on a whim; maintaining them (at least, if they become successful) can require significant effort, and institutions need to make rational choices of supported functionality, languages and authentication mechanisms.

Linked Data is a conceptual and technical approach to data storage and access, which is supported by several standards, including RDF and SPARQL. Linked Data can make it easier to integrate systems or use data across silos and technologies. However, it is not the only valid way to share data.

With the explosion of developments often called the "NoSQL Movement," new data storage technologies have become available or have matured to a point where they are usable. In this "NoSQL Movement" it is difficult to generalise about the approaches taken. However, these developments are often defined by what they are not: they are not relational databases and do not adhere to some traditional RDBMS constraints. NoSQL databases make some trade-offs against the behaviour of relational databases, typically gaining performance and flexibility at the cost of reducing reliability and ease of use.

Cloud computing also presents new ways of buying computational and storage capabilities as services. For the broader concepts of a data-centric architecture, moving to "cloud" does not significantly change the data model or architecture of an organisation (although it might change the system and responsibility boundaries). The potential advantages derive from the business model, making new capabilities cheaper and more flexible than would otherwise be the case.

Priorities for institutions are:

1. Take interface design seriously at the design/procurement stage. This might include developing your own APIs, or opting to consume external data.

2. Think about internal data sharing first: sharing does not have to be open or external to be beneficial.

3. Consider APIs as a route to data – not a way to control software.

4. Consider whether to standardise on RDF, RESTful APIs where appropriate, or something else.

5. As ever, beware the hype of these new technologies. Linked Data, the Cloud, and NoSQL databases can all be strongly beneficial – but only in the right systems.

In summary, the principles underpinning data-centric architectures are strong and – if applied with care – will lead to the development of systems of systems that are easier to maintain, adapt, and extend. These principles should feed into the planning of all enterprise systems. The specific technologies discussed in this paper provide a useful suite of capabilities, which should be considered as options when building or purchasing systems.

Table of Contents

1. Introduction: the need for improved data management

Managing corporate data is a task that has occupied and frustrated organisations of all sizes. They understand that they have access to data of critical importance for business operations and strategic planning, but are unable to exploit these data. The CEO of HP, Lew Platt, famously once said: “If only HP knew what HP knows, we would be three times more productive.”

One concept in systems design that is growing increasingly important is termed data-centric architecture. This approach to building and buying systems focuses primarily on the institutional data (as opposed to systems) and is intended to facilitate the sharing of data between processes requiring shared data.

Further and Higher Education (FHE) institutions have particular challenges, given that they often have very broad data assets covering administrative, teaching, and research activities. Funder mandates and other compliance requirements are increasingly forcing institutions to tackle the challenge of managing data. In addition, institutions have a broad range of external data-sharing needs to meet, including:

- mandatory requirements (such as providing employee information to tax authorities, preparing Key Information Sets (KIS), making submissions to the Research Excellence Framework (REF), etc);
- community engagement (for example, sharing emerging research findings with collaborators).

Different institutional and technical architectures have been used to control the torrent of information that organisations collect, with varying degrees of success. The wholesale move to electronic information systems and the more recent move to fully Internet-based and interconnected information systems have exacerbated the problems.

We are now operating in the era of 'big data', where a combination of factors has enabled us to deploy new methods of managing and exploiting relatively large and dynamic datasets. These factors include (White, 2011):

- new systems that are generating data in new ways (e.g. social media);
- improved analytical capabilities;
- operational business intelligence that can turn analytic outputs into action;
- faster hardware and cloud computing services, which make possible larger-scale data processing.

Together, these technologies allow more data to be gathered and that data to be processed in more depth, more quickly, and more economically than was previously the case. However, to exploit them, it is necessary to look above the technologies to the overall data architecture.

One concept in systems design that is growing increasingly important to support these factors is what is termed data-centric architecture. This approach to building and buying systems focuses primarily on the institutional data (as opposed to systems) and is intended to facilitate the sharing of data between processes requiring shared data. A fully fledged data-centric architecture requires a level of technical maturity that may be challenging for some organisations and could be expensive and high-risk to implement quickly. There are, however, pragmatic steps that HE and FE institutions can take now to begin moving toward a more data-centric approach, and these steps are the focus of this report.

The remainder of the report is structured as follows:

- Section 2 describes how institutions share data (both internally and externally) and then describes data-centric architectures in more detail, considering how a *pragmatic* data-centric architecture could meet the needs of institutions. This section will be of interest to all readers concerned with how institutions can respond to increasing challenges in data management and share data more effectively.
- Section 3 describes three key technologies that can support data-centric architectures and 'big data' challenges: APIs, Linked Data, and NoSQL databases. This section contains technical descriptions of most interest to those responsible for strategic technology choices. Technical readers unfamiliar with these technologies may also find this section useful.

- Section 4 sets out the author's view of the role that key concepts and technologies will play in the HE sector over the next five years. This section will be of interest to all readers concerned with the ability of organisations to meet increasing requirements for managing and sharing data efficiently in the era of big data, cloud computing, and innovative data-centric architectures.

2. Data-centric architectures

Data sharing

This section explains some requirements for HE and FE institutions sharing data internally and externally, and then describes an architectural concept that can allow institutions to meet the usage scenarios described.

All organisations are required to share data in some ways – whether internally, to a defined community, or openly. Data sharing is often held to be a benefit in its own right – in particular within the education and research communities – but there are strong and increasing drivers to share more, and more openly (DeSantis, 2012). It is important to recognise, however, that the costs and benefits of data sharing are not aligned: the costs are borne by the data source, while the benefits are enjoyed by the data consumer.

Where the creation and use of data both take place internally, this split between costs and benefits is not an issue since creators and consumers are part of the same enterprise. Notwithstanding details of budget lines etc, it is clear to see how the organisation as a whole benefits from the support for the easier internal access to its own data.

Where the consumer is external to the organisation, the benefits of sharing are likely to be indirect, but they may still be strong. These benefits may be driven by factors such as:

- mandatory requirements (such as providing employee information to tax authorities);
- community engagement (such as sharing emerging research findings with collaborators);
- public good (most institutions are publicly funded, and believe that sharing data is part of their mission).

A previous JISC report focusing on Linked Data discusses the benefits, costs, and risks of data sharing in some detail, and will be of interest to readers who are considering the business case for data-centric architectures and other data-sharing approaches (Hawtin, et al., 2011). The technologies for sharing data are similar for internal (enterprise) and open data, but the drivers differ.

Enterprise data sharing

Institutions have adopted a wide variety of approaches to creating data architectures internally. In many cases, these approaches are developed *ad hoc*: data flows are created between systems as needed, and may rely on tightly coupled interfaces, or even manual processing (e.g. entering details of grants awarded into both a finance system and a research management system).

A few institutions have begun using open data approaches to create frameworks of institutional data.[5] These institutions appear to have focused primarily on open data, which is an easier place to start, as there are (by definition) no difficulties with authentication or authorisation of users.

The institution with the most wide-ranging set of data available is probably the University of Southampton.[6] This site is an attempt to liberate data from its existing silos in a very pragmatic way. Data contained are all either in the public domain already, or would be disclosed under a Freedom of Information request, thus avoiding personal or otherwise sensitive data. A full case study is contained within a recent JISC report (Hawtin, et al., 2011, pp. 38-40), but the key points of the Southampton experience are:

- It was undertaken in a "softly-softly" approach; there was no attempt to undertake any wholesale re-engineering.
- Data are ingested from diverse internal data sources, in whatever format is possible, ranging from *ad hoc* uploads of spreadsheets to automatic data transfers.
- Although RDF / Linked Data (see explanations within section 3: 'Tools and technologies') was the particular technology selected for data publishing, the focus was on making data available, rather than linking them.

[5] A number of registries of known institutional data sites exist, including <http://hub.data.ac.uk/> and <http://linkeduniversities.org/lu/>. Specific university datasets are often listed at <http://thedatahub.org/>.

[6] http://data.southampton.ac.uk/

An interesting observation from the JISC report was that, although Southampton had focused on open data, the benefits described were internal. Southampton would have received these benefits without making its data public (Hawtin, et al., 2011, p. 16).

A counterpoint to the focus on open data is provided by the University of Lincoln, which has developed a range of APIs, including the ability to deliver internal data using authenticated APIs (see examples within section 3: 'Tools and technologies').

A range of stakeholders in the HE sector require data from institutions, including most obviously HESA, UCAS, and the HEFCE REF, but there are also several hundred other types of reporting in the sector.[7] The data provided to these organisations are typically fragmentary, and have consequently caused institutions to develop *ad hoc* systems to meet these requirements.

Within the FE sector, there is a more structured, sector-wide approach to data management, with the Information Authority[8] established "to set data standards and govern data collection and use for further education and training provision in England", and the associated entity, The Data Service.[9] Although analogous to HESA, the Information Authority appears to have a more comprehensive view of the information environment in FE.

HESA, however, recently completed a major project ("Project B")[10] to redesign the information landscape for HE, in order to meet the needs of the wider group of users, reduce the duplication that currently exists, and provide timelier and more relevant data. This project incorporated input from a wide range of stakeholders and reported in June 2012. The recommendations have been accepted and will likely shape the future data environment for HE, albeit with the collaborative approach recommended likely to lead to a gradual and incremental change.

Research and teaching data sharing

Beyond these enterprise uses of corporate data, institutions are sharing data in a range of ways to support their research or teaching activities. This kind of data sharing is driven more by the culture of academia than by the potential benefits to enterprises, and the types of data handled are more diverse. Datasets are likely to be 'multi-structured', rather than the defined set of

[7] http://landscape.hesa.ac.uk/hebrg-survey-of-statutory-and-external-returns-help-us-to-help-you/

[8] http://www.theia.org.uk/

[9] http://www.thedataservice.org.uk

[10] http://landscape.hesa.ac.uk/

fields and formats that tend to define corporate datasets. Different research projects and teaching systems can each produce a wide range of data types, often with "unknown, ill-formed or overlapping schemas." (White, 2011, p. 1)

The management of research data is a significant and growing challenge for institutions. Driven by a range of factors including strong research council mandates, many institutions are in the process of developing infrastructures to support the research data lifecycle. Lessons from these developments are still emerging, and integrated information research data management systems are typically deployed at limited scale or scope,[11] and such projects frequently run into difficulties with accessing data. Here is one of many examples (Ferguson, 2011):

> As well as getting access to data on research applications and data from the research repositories, the team had been hoping for access to data on research grants, holders etc direct from university's finance system. This was not forthcoming. They were also hoping for access to well-structured data from research councils – such data of sufficient quality was also not available in practice and the team were forced to obtain "screen-scraped" data from research council web sites and online databases.

Tellingly, however, that project to which Ferguson refers was successful despite the considerable problems in accessing data. This is a clear demonstration of the demand within organisations for effective exploitation of their data. One individual is quoted by Ferguson:

> Initial discussions with Research and Enterprise Development in the University indicated that there was limited scope in existing systems to draw together, integrate and gain different views of data relating to research, research outputs, research income and researchers themselves. [This project] has demonstrated how this could happen far more systematically and the potential benefits in research intelligence, benchmarking, promotion and management/enhancement of research performance. For a University that describes itself as 'research-intensive', access to and manipulation of these data at individual, organisational unit and university-wide levels is very important. This is further evidenced by the University now taking forward options analysis and business case for a university-wide system.

[11] http://www.jisc.ac.uk/whatwedo/programmes/di_researchmanagement/managingresearchdata.aspx

The paradigm of data-centric system design

This section introduces some abstract concepts of system engineering and of data management as an introduction to the remainder of the report, which has a more technological focus.

Information systems have typically been constructed as sets of services, which are more or less strongly coupled. This service-centric paradigm works well in many cases, such as in those where services are closely coupled to provide integrated functionality. However, many datasets now being collected and analysed are different: value resides in the data rather than the service, and the service only exists to expose the data. For such systems, a data-centric design paradigm is often more appropriate. This paradigm treats the data and metadata as the key aspects within the system, and hides component behaviour. The system is defined by its data model, with producers and consumers of data acting within that model.

Implementing this kind of architecture requires a fundamental reconsideration of the relationships between systems. Although data-centric architectures are the subject of this paper, there is no single definition of what they are. In a robustly data-centric design, data handling is managed by a data bus, typically using the Data Distribution Service for Real-Time Systems (DDS) standard (The Object Management Group, 2012), which represents the "World Model", and this data bus is responsible for establishing and managing data flows between components according to agreed data-handling contracts (Joshi, 2011). Implementing this kind of system is a major software-engineering task, beyond the scope of this paper. However, the key concepts are perhaps more applicable within the HE/FE context, which can allow improvements without wholesale re-engineering:

- **Expose the data and metadata**. Information is the priority, and without exposing this information it is clearly impossible to exploit it!
- **Hide component behaviour**. Rather than concentrating on the functional interface between systems, each interface should operate on data: it should ask for (or update) the data and not refer to the state or operation of the component behind it.

This is similar to the situation for Service Oriented Architectures (SOA), where large enterprise systems are often created with an Enterprise Service Bus (ESB), but many benefits have nonetheless been realised in a more lightweight manner – for example, by linking discrete systems with data-centric APIs (see section 3 below). These “Pragmatic” data-centric architectures, and the tools to support them, will be the focus of this report.

Priorities for institutions

Two key systems approaches have the potential to support better institutional use of data: 1) designing Systems of Systems (SoS), and 2) establishing coherent data management policies and practices. These strategic design decisions, when supported by appropriate technologies, can go a long way toward reducing the silos of information that are common in institutions.

Managing systems of systems

To state the obvious, engineering software applications and architectures is a challenging, specialist task – and architectural design decisions sometimes come down to the experience of the team involved, the time and effort available, and even personal preference.

When creating software, there is always the temptation to create tightly coupled systems in which components are intricately linked with each other. Software engineering for complex systems now often applies the concept of a System of Systems (SoS). Much of the early work on SoS was undertaken within the defence context (Sage & Cuppan, 2001), but the characteristics taken to define SoS (Maier, 1998) are very recognisable in HE and FE today:

- operational independence of the individual systems;
- managerial independence of the systems;
- geographical distribution;
- emergent behaviour;
- evolutionary development.

Of these characteristics, emergent behaviour is the most abstract. This term describes actions “that cannot be localized to any single component of the system but instead produce effects… that arise from the cumulative action and interactions of many independently acting components” (Fisher, 2006). Emergent behaviours can be negative (traffic is slow at certain times of day not due to the actions of any individual driver, but due to the overall system) or positive (email is delivered not by any one system, but by a combination of client, mail servers, routers, DNS, physical links etc).

We can note an associated set of emerging good-practice approaches to designing and engineering systems of systems (Office of the Deputy Under Secretary of Defense for Acquisition and Technology, 2008, pp. 21-23):

- addressing organisational as well as technical issues in making system-engineering trade-offs and decisions;
- acknowledging the different roles of systems engineers at the system versus the SoS level as well as the relationship between the system engineering done at the two levels;
- conducting balanced technical management of the SoS;
- using an architecture based on open systems and loose coupling;
- focusing on the design strategy and trade-offs both when the formal SoS is first established and throughout the SoS evolution.

Many of these factors are out of scope for this paper, but open systems and loose coupling are in scope (see below).

Establishing coherent data management

One of the key concepts in a data-centric architecture is that there is a common World Model consisting of a single "true" picture of the organisation's information. In order to create a shared world view, it is necessary to have systems in place (both technical and organisational) to enable and then enforce compliance.

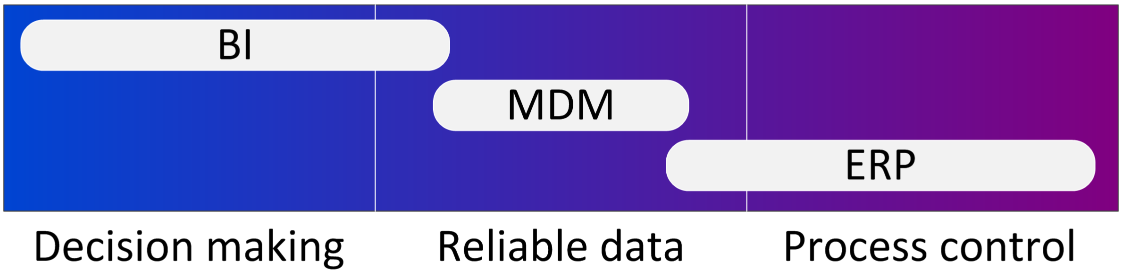
Master Data Management (MDM) is the process of managing this world view, and it is a hard problem to solve (MIKE2.0 Methodology, 2011b). MDM processes should cover the data lifecycle, including:

- identifying data sources internal and external to the organisation;
- collecting and enriching data;
- data transformation, schema mapping and normalisation;
- rule administration, error detection and correction;
- consolidating and storing data;

- data classification, including taxonomy control;
- distribution of the master data to data consumers.

There are soft boundaries between MDM, Enterprise Resource Planning (ERP), and Business Intelligence (BI) solutions. Each attempts to understand the organisation holistically: crudely, MDM forms a consistent view of the information within the organisation; ERP is focused on making sure that business processes and associated data work together; and BI exploits the data within the organisation to make or support business-relevant decisions. Naturally, ERP requires some data coherency across the organisation, so ERP approaches include data management, but not to the same depth as MDM.

Business Intelligence, Master Data Management, Enterprise Resource Planning



A range of tools is available to support enterprise MDM, including SAP NetWeaver MDM, Oracle MDM, IBM InfoSphere MDM, Informatica MDM, and many others. Within the HE domain, specific ERP systems such as Agresso, CampusVue, and Jenzebar may prove to be appropriate starting points. Implementing any of these systems requires a significant level of architectural maturity, and quite likely an extensive technology and business-process change programme, which are out of scope for this report. However, readers may find the approach from the MIKE2.0 standard (MIKE2.0 Methodology, 2011a), or PriceWaterhouseCoopers (PriceWaterhouseCoopers, 2010) to be useful starting points.

Adopting a practical approach to data-centric architecture

Adopting a data-centric approach is not a binary choice. There is a range of options available to institutions with a range of benefit, cost, and risk profiles. Different institutions will make different choices based on their own circumstances.

There are already strong drivers on institutions to improve their data management, and to take a more formal approach to this. This may be through the wholesale adoption of MDM, or through a gradual process of developing modular and interconnected systems in future, and identifying authoritative sources for particular classes of data (and then insisting that new systems re-use these sources). Both approaches – if implemented with care – would likely lead to benefits in terms of exploiting organisational data as well as reducing the risks to confidentiality and integrity of datasets.

There are potential downsides and risks to changes in technology or technology strategy, and the most significant for data-centric architectures is that systems of systems can be brittle: a failure in one component can cause the entire system to fail. However, it is likely that this brittleness already exists in existing *ad hoc* arrangements of systems, perhaps without having been formally considered.

3. Tools and technologies

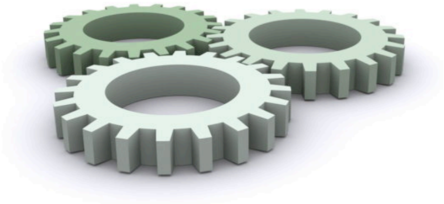
Overview of APIs

An API is an Application Programming Interface: the channel through which software components communicate with each other. For the purposes of this report, APIs are the links which hold together the systems within a pragmatic data-centric architecture.

A forthcoming JISC report will cover the use of APIs in detail; this section will introduce some concepts, and outline some of the most important factors to consider.[12]

HE and FE institutions can choose from a suite of tools and technologies available to meet different use cases within different usage scenarios. This section introduces the key concepts of APIs, Linked Data, NoSQL, and cloud computing, giving examples of usage and comparing the pros and cons of each technology.

APIs have been described in many ways, and *APIs: A Strategy Guide* contains a particularly clear statement (Jacobson, et al., 2012): "Technical definition: An API is a way for two computer applications to talk to each other over a network (predominantly the Internet) using a common language that they both understand."



APIs follow a specification, meaning:

- The API provider describes exactly what functionality the API will offer.

- The API provider describes when the functionality will be available and when it might change in an incompatible way.

[12] For information about this forthcoming JISC work on 'Digital Infrastructure Directions Report: Advantages of APIs' see http://www.jisc.ac.uk/fundingopportunities/funding_calls/2012/02/did_advantages_of_api.aspx. A previous JISC report (Guy, 2009) is now rather dated. The O'Reilly publication *APIs: A Strategy Guide* (Jacobson et al., 2012) provides an excellent overview of current good practice, and this is recommended reading for anyone considering developing an API.

- The API provider may outline additional technical constraints within the API, such as rate limits that control how many times a particular application or end-user is allowed to use the API in a given hour, day, or month.
- The API provider may outline additional legal or business constraints when using the API, such as branding limitations, types of use, and so on.
- Developers agree to use the API as described, to use only the APIs that are described, and to follow the rules set out by the API provider.

In addition, the API provider may offer other tools such as:

- mechanisms to access the API and understand its terms of use;
- documentation to aid in understanding the API resources (such as example code and developer communities to support those using the API);
- operational information about the health of the API and extent of its usage.

Note: Remember that the structure of the API is part of the contract. The contract is binding, and it cannot be changed casually.

When implemented properly, an API represents a system boundary. A piece of software that depends on (consumes) an API should not require knowledge of how the software that publishes the API works – only that it does, and that the API behaves in a predictable and well-defined way. This has significant benefits in engineering systems of systems. These benefits may include: greater reusability of software systems, greater flexibility in reconfiguring systems, and reduced overheads when creating new systems.

APIs may be internal (to the organisation) or external, may require authentication or may be open, or may provide different functionality to different user groups.

As with any development activity, it is fundamentally important when considering where and how to create APIs to consider who the end-users will be, who will be creating applications using your API, what use cases you expect to meet, and in which usage scenarios.

APIs should not be created on a whim – maintaining them (at least, if they become successful) can be a significant activity, and rational choices of supported functionality, languages and authentication mechanisms are necessary.

Understanding the users has a further implication that runs deeper than selecting functionality. It is necessary to decide upfront what approach you will take to maintaining the API in future: will it be version-controlled or versionless?

- **Version-controlled APIs** allow the client to call a specific version. This allows total stability for clients, at the cost of the API provider having to maintain multiple versions.
- **Versionless APIs** allow the API to evolve without creating new versions. To deliver the stability that developers require, this approach requires very careful thought. It is easier to add features later than to remove or change them, and and as such it is often better to withhold features – even potentially very useful ones – if you cannot guarantee long-term stability.

Many of the potential risks of using a versionless API are mitigated when there is a smaller development community consuming the API, as is often the case for an internal API.

API technologies

SOAP and REST

Web APIs are a particular class of API, using the infrastructure of the Web to transfer information between systems. Of particular interest are so-called "RESTful" APIs (REpresentational State Transfer), which are especially suitable for the transfer of information between loosely coupled systems – which is an ideal fit for the concept of a pragmatic data-centric architecture.

For more tightly coupled systems, or those that require particular security functionality (typical of enterprise systems), it may be more appropriate to use the more powerful, more general, and more complex Simple Object Access Protocol (SOAP) approach. Many of the same design considerations apply regardless of which approach is adopted.

REST APIs (in their purest form) are implemented so that the Standard HTTP "verbs" (primarily GET, POST, PUT and DELETE, but other verbs may be used as well) operate on resources that are themselves identified by unique URI patterns.

Typically, this kind of RESTful URL schema would look something like this:

```
https://hypothetical-sp.com/customer/Alice

https://hypothetical-sp.com/customer/Bob

https://hypothetical-sp.com/customers/
```

In this example, /customers/ is the collection of all customers. A new customer would be added by POSTing the new object representing them to the /customers/ collection.

It is important to distinguish between REST and other non-SOAP patterns. For example, Amazon offers a range of APIs to control their AWS "cloud" provision. Alongside a SOAP API, they offer a "Query" API, which uses GET or POST requests with appropriate parameters to control instances. A typical request of the Query API might look like this:[13]

```
GET https://ec2.amazonaws.com/?Action=StopInstances&InstanceId.
1=i-10a64379&AUTHPARAMS
```

which instructs the AWS fabric to stop the instance identified as "i-10a64379". If this were pure REST, the request would instead have a structure such as:

```
DELETE https://hypothetical-ws.com/instance/i-10a64379/
```

with the authentication parameters passed in the HTTP headers.

This is not to say that Amazon's API is necessarily incorrect or inappropriate – Amazon instances are very complex, stateful objects – but rather to emphasise the point that REST requires both the object identifiers and the methods of controlling them to share the philosophy. Developing a consistent schema for unique identifiers for data objects is a key task for a pragmatic data-centric architecture (see below).

Data structures: XML and JSON

APIs can be designed to return data in a range of formats, of which XML (eXtensible Markup Language)[14] and JSON[15] (JavaScript Object Notation) are the most widely used. XML is powerful, extensible, has an enormous range of available ontologies, but requires complex parsing by the client. JSON objects are much easier to incorporate into a wide range of client software without the user (i.e. the developer) having to create custom parsing code, and these JSON arrangements are sufficient for most APIs (Jacobson, et al., 2012, p. 65).

[13] This example interaction is from http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/ApiReference-query-StopInstances.html

[14] http://www.w3.org/XML/

[15] http://json.org/

Examples of API usage in HE and FE

Whereas there are many examples of simple exploitation of a Google Maps API to display data on a map, there are few examples of the use of open APIs for enterprise systems within Higher and Further Education. The majority of existing examples are built into "resource discovery" systems: for example, in library management systems consuming Z.39.50 interfaces from a range of sources including Copac,[16] Zetoc,[17] SUNCAT[18], and other libraries.

The University of Lincoln has created a range of APIs for specific systems such as their 'Nucleus' endpoints for Calendars, Events, People and Print services,[19] and the datasets provided through these APIs are used to drive sections of the University's corporate Website and phone directory (Bilbie, 2012), albeit sometimes with intermediate parsing.[20]

The JISC Monitoring Unit provides another example of API usage with its database of organisations and services, to which it provides a REST API for access to data.[21] As explained on the Web page introducing this API, "The Monitoring Unit's web services are loosely based on the concept of Representational State Transfer, commonly known as REST. Data about organisations is accessed at specific URIs (much like visiting a web page) and there is some choice in the media you may request – currently XML or JSON (the default is XML)."[22]

A current JISC project[23] is also identifying the range of APIs available from JISC-funded resource discovery services, and how they currently interact. As explained in this project's preliminary statement of objectives, "The purpose of this work is to produce an overview of the data sets that underpin the various elements of JISC's digital infrastructure. The overview will be used to identify gaps and efficiencies in the digital infrastructure and provide higher education innovators with information on datasets that are relevant to the systems they are using or developing."

[16] http://copac.ac.uk/developers/

[17] http://zetoc.mimas.ac.uk/about.html#openurl

[18] http://www.suncat.ac.uk/support/z-target.shtml

[19] https://github.com/unilincoln-ost/Nucleus-Docs/wiki

[20] http://httpster.org/the-future-of-data-lincoln-ac-uk/

[21] http://www.jiscmu.ac.uk/api/services

[22] http://www.jiscmu.ac.uk/api/

[23] For an initial overview of this current 'data audit' work see http://www.jisc.ac.uk/fundingopportunities/funding_calls/2011/11/dataaudit.aspx

Linked Data

Linked Data is a conceptual and technical approach to data storage and access, which is supported by several standards. A recent JISC report considered the business case for Linked Data and presented a range of case studies from HE, government and industry (Hawtin, et al., 2011). This report is recommended reading for examples of Linked Data in use, and its key findings may be paraphrased as follows:

- Linked Data technologies are still maturing, which hinders development of best-practice advice and obscures evidence of the benefits.
- There is no single correct way to implement the ideas and ideals of Linked Data. The specific technological decisions made in each case may well be different, and these decisions are affected to a significant degree by the mix of existing skills and systems within the institution.
- It is possible to construct plausible arguments that real business benefits to HE and FE institutions could be achieved; however, it is not yet possible to build a concrete business case on these arguments.
- Institutions are the biggest users of their own data, and it is important that the term 'Linked Data' is not used synonymously with 'open data', since this may both confuse and discourage potential adopters. Linked Data need not be open.

At a technical level, the following benefits of Linked Data were identified:

- It is easier to integrate systems or use data across silos and technologies.
- The data structures of Linked Data (or ontologies) are flexible and extensible in the event of future change.
- These factors make it easier for developers to work across systems.
- These factors also support flexibility in data management and use.

These benefits are based on the following technical characteristics of Linked Data:

- RDF data is 'easier' to pass between systems in that it is well suited for use as a common and extensible "translation" interface layer;
- a single SPARQL endpoint can interrogate multiple datasets (as seen in data.gov.uk and in Southampton Open Data Service at http://data.southampton.ac.uk/ etc).

Linking data: RDF and SPARQL

The term Linked Data was coined by Sir Tim Berners-Lee, which he has embodied in four principles (Berners-Lee, 2010):

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information using the standards (RDF, SPARQL);
4. Include links to other URIs so that they can discover more things.

The first two points will be familiar from the discussion of REST APIs above. What is different is that URIs need not resolve: the URI may represent a namespace rather than a Web page, as for XML[24].

A detailed description of the technologies of Linked Data is beyond the scope of this report. A list of relevant textbooks is maintained by the W3C,[25] and the remainder of this section will introduce the key elements: RDF and SPARQL.

The most common format for Linked Data is RDF (Resource Description Framework),[26] which represents information as triples of the form <subject> <predicate> <object>. For example, say a Web page had the following metadata:[27]

```
location:      http://www.example.org/index.html

creator:       John Smith

creation-date: August 19, 1999

language:      en
```

[24] This is a subtle distinction. As an example, doi:10.1234/123 is a URI that refers to a particular type of digital object – but it is not possible actually to open that object in a Web browser, for example – it is just the reference. http://dx.doi.org/10.1234/123 on the other hand is a URL that can resolve that DOI.

[25] http://www.w3.org/2001/sw/wiki/Books#Textbooks

[26] http://www.w3.org/TR/rdf-primer/

[27] This example is taken directly from http://www.w3.org/TR/rdf-primer/#statements

In RDF, this information could be represented as the following series of triples:

http://www.example.org/index.html	subject
http://purl.org/dc/elements/1.1/creator	predicate
http://www.example.org/staffid/85740	object

http://www.example.org/index.html	subject
http://www.example.org/terms/creation-date	predicate
August 16, 1999	object

http://www.example.org/index.html	subject
http://purl.org/dc/elements/1.1/language	predicate
en	object

There are two key points to note here:

1. The predicates are also described as URIs: they represent elements in an ontology (and these elements are the logical frameworks necessary to describe these relationships).

2. The object of the first triple is a link: it refers to the object that represents John Smith. Much as a well-normalised relational database operates by reference,[28] RDF allows linking between different classes of data and these datasets can be held by different organisations and in different systems. A particular capability of a Linked Data approach is to link disparate data about the same entity to establish an equivalence; this relationship is known as sameAs. This ability is very helpful in breaking information out from silos (for example, the institution identified by HESA code 0178 is the same institution that has the domain name bangor.ac.uk, and a sameAs relationship can link these two entities). It is important to bear in mind that,

[28] For example, one would not repeat a customer's contact details in a table containing bill details; instead, a foreign key from the table of customers would be included.

> although the SPARQL interface may be common, it is in practice still necessary to have a detailed understanding of the data provided through it.[29]

SPARQL

RDF triples are, unsurprisingly, stored in triplestores. Much as relational databases typically have SQL interfaces, the standard query language for RDF data is SPARQL (pronounced 'sparkle').[30] Essentially all triplestores offer SPARQL interfaces, and many data services offer SPARQL APIs. SPARQL maintains the graph structure of triples, and can be used to conduct inferencing queries[31] (provided that the system underpinning it is capable of undertaking the inferencing).

Non-relational databases (NoSQL)

Over the last few years, new data storage technologies have become available or matured to a point where these technologies are now usable. This explosion of concepts is often called the 'NoSQL Movement,' and it is difficult to generalise about the approaches taken, but they are often defined by *what they are not*: NoSQL technologies are not relational databases using the RDBMS paradigm.

In "traditional" relational databases that hold most information systems together, the data schema is rigidly fixed and changes typically require careful thought and planning; there is a risk that a schema change will break applications. Nevertheless the RDBMS paradigm has significant strengths: it is well understood in theory and practice, and a wide range of products are available with mature supporting ecosystems.

By contrast, NoSQL databases make some trade-offs against the behaviour of relational databases – typically gaining performance and flexibility at the cost of reducing reliability and ease of use. These performance and flexibility gains are usually achieved in two ways:

- **Very flexible schemas**: NoSQL databases are often termed "schemaless", but this is not strictly true; clearly, there must be some structure to the data to enable client code

[29] This is analogous to SQL – although the language may be common, it is necessary to understand the underlying database structure before it is possible to write meaningful queries.

[30] http://www.w3.org/TR/rdf-sparql-query/

[31] Inferencing is the ability to take triples such as <humans> <are> <mammals>, and <mammals> <have><lungs>, and be able to answer a question of whether humans have lungs, despite there being no explicit triple in the store stating <humans><have><lungs>.

to access it. What is different is that these schemas need not be defined in advance, and can be easily modified. Put another way, responsibility for understanding the data structure is transferred at least partly to the application layer using the data.

- **Horizontal scalability**: databases that can scale through the addition of more servers, rather than through the increase in the power of the existing server. Although most relational databases can be effectively sharded,[32] many NoSQL databases have been designed from the outset to operate in a distributed manner.

This trade-off has been expressed well by Loukides (2012):

> The ACID properties (atomicity, consistency, isolation, durability) have been drilled into our heads. But even these come into play as we start thinking seriously about database architecture. When a database is distributed, for instance, it becomes much more difficult to achieve the same kind of consistency or isolation that you can on a single machine. And the problem isn't just that it's "difficult" but rather that achieving them ends up in direct conflict with some of the reasons to go distributed. It's not that properties like these aren't very important – they certainly are – but today's software architects are discovering that they require the freedom to choose when it might be worth a compromise.

In practice, most production environments that have adopted NoSQL also use a complementary RDBMS arrangement; it has been suggested that NoSQL could be better thought of as "NotOnlySQL" (Warden, 2011, p. 5). Even Oracle, which has in the past published a well-researched criticism of the NoSQL approach (Oracle, 2011), now offers a NoSQL product.[33]

NoSQL technology types

Most "NoSQL" databases fall within two classes (Warden, 2011):

- **Document stores**. In this paradigm, an element of data is a document, which is analogous to a relational database row. These documents may be grouped into collections (analogous to RDBMS tables), but other organisational structures are available. What is distinctive is that (unlike in RDBMS tables, where fields are defined and static) documents in document stores can have any arbitrary set of fields.

[32] Sharding is a technique used to scale relational databases horizontally. It typically involves dividing the data into segments (for example records for students with surnames beginning A-J, and those beginning K-Z), and hosting each segment – known as a shard – on a separate server.

[33] http://www.oracle.com/technetwork/products/nosqldb/

- **Key-value stores**. These systems store key/value pairs, and (in their purest forms) provide just three primitive operations: Get data associated with a key, store data against a key, and delete a key and its data. These systems typically provide predictable performance characteristics, at the cost of complex operations (such as query building) becoming the responsibility of the application.

There are a large number of products now available in this space, and attempting to list and compare them all here would be both uninteresting and rapidly outdated. Products of particular interest to the Higher and Further Education audience at the time of writing include: MongoDB,[34] CouchDB,[35] Cassandra,[36] DynamoDB,[37] SciDB.[38]

Overview of some NoSQL database products

Name	Class	Notes
MongoDB	Document store	Stores and interfaces through JSON objects, particularly suited to large datasets, relatively low-maintenance. Supports automatic sharding.
CouchDB	Document store	Stores JSON objects, no built-in horizontal scalability, treats query responses as views which are themselves documents. Takes a multiversion concurrency approach.
Cassandra	Key-value store	Complex to start with, but delivers a lot of power and flexibility.
DynamoDB	Key-value store	Available through Amazon Web Services, likely of use when other components of the infrastructure are also hosted there. Focus on high and predictable performance.

[34] http://www.mongodb.org/

[35] http://couchdb.apache.org/

[36] http://cassandra.apache.org/

[37] http://aws.amazon.com/dynamodb/

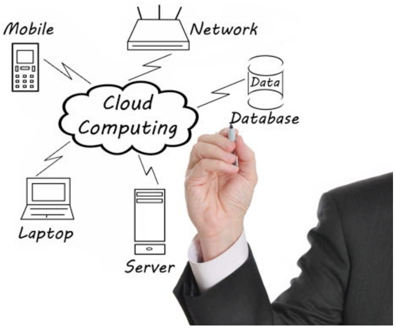
[38] http://www.scidb.org/

Name	Class	Notes
SciDB	Multidimensional arrays	SciDB is neither a document store nor a key-value store, and is designed for storing very large scientific datasets (up to hundreds of petabytes). May be particularly relevant for research data.

In the past, choosing between RDBMS options (such as between MySQL and PostgreSQL) for a particular Web application was relatively straightforward: the technologies were similar, and decisions came down to specific features. NoSQL changes the game – one size no longer fits all (Clarke, 2010). Loukides (2012) summarises these technology choices well:

> They're the same issues software architects have been dealing with for years: you need to think about the whole ecosystems in which the application works; you need to consider your goals (Do you require high availability? Fault tolerance?); you need to consider support options; you need to isolate what will change over the life of the application, and separate that from what remains the same. The big difference is that now there are options; you don't have to choose the relational model.

New models of cloud computing



Alongside the technical and conceptual developments discussed above, there has been a change in mechanisms of providing and using computing power and storage. Within the model known as 'cloud computing', the fundamental idea is that rather than making capital investments in hardware and software, elements can be outsourced for provision as services.

The National Institute of Standards and Technology (NIST) defines cloud computing as follows:

> A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. (Mell & Grance, 2011)

This NIST definition goes on to define essential characteristics of having on-demand self-service, broad network access, resource pooling, rapid elasticity and a measured (i.e. pay-as-you-go) service.

The key implications of this approach are:

- Capital expenditure is transferred to become operational expenditure.
- Clouds are elastic, and systems can be scaled “on demand”. It is possible to add capacity to most cloud services on a timescale of just minutes. This can permit very responsive systems design and eliminate investment in over-capacity.
- As the cloud datacentres are operated on a very large scale, this should bring efficiencies, reliability and lower prices for customers.

There are three common service models for cloud: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

Most research into using cloud services within Higher and Further Education has focused on IaaS provision to supplement or replace institutional research compute clusters (Hammond, Hawtin, Gillam, & Oppenheim, 2010) and has demonstrated that cloud costs can be on parity with the best institutional provision (Hawtin, et al., 2012). These studies suggested that undertaking data-intensive research on the cloud would be uneconomic: for example, per-GB storage charges are high compared to those incurred by the purchase and operation of a portable 1-parTB drive stored in a researcher’s desk. The cloud providers deliver highly reliable storage, whereas researchers tended to require “good-enough” storage. When considering the management of data at large scale, however, it may simply be impossible to handle locally, which makes cloud provision a new capability, rather than just an extended capacity.

Data ‘in the cloud’

The first, and very important, point is that there is no “cloud”. Cloud computing actually means that data and services are provided from and hosted at a service provider’s datacentres. That said, such datacentres are typically multi-billion-pound facilities providing far greater capability, redundancy and resiliency than any institution can deliver locally.

When considering data “at scale”, the challenge is not simply storing the data; the challenge is making datasets accessible to analytical resources that can process them. What this means in practice is developing analytical algorithms that scale horizontally, and deploying them into a parallel-processing framework.[39]

[39] In other words, large tasks are broken down into smaller component tasks, which can be distributed amongst a large number of worker nodes, before the answers are collated.

One such parallel-processing framework is MapReduce, developed by Google to power its search engine (Dean & Ghemawat, 2004). Hadoop[40] is a mature and well-supported open-source implementation of this parallel-processing framework, available to be installed and operated on local clusters (albeit with some degree of difficulty).

Several cloud vendors also provide MapReduce as a service: some examples include Google itself within the App Engine platform;[41] Microsoft in Project Daytona (which is focused specifically on researchers[42]) as well as a developmental enterprise Big Data solution;[43] and Amazon Elastic MapReduce (EMR).[44] All these applications may be integrated with the data stores managed within the provider's platform. EMR is particularly interesting due to its tight integration with DynamoDB (see above) and Amazon S3 (a well-understood storage system with a mature ecosystem of tools).[45]

A wide range of materials describing MapReduce/Hadoop is available, including a free ebook produced by IBM (Zikopoulus, et al., 2012) that explains in detail these open source technologies (before discussing IBM-specific products)[46] .

[40] http://hadoop.apache.org/

[41] http://code.google.com/p/appengine-mapreduce/

[42] http://research.microsoft.com/en-us/projects/daytona/default.aspx

[43] http://www.windowsazure.com/en-us/home/scenarios/big-data/

[44] http://aws.amazon.com/elasticmapreduce/

[45] http://aws.amazon.com/s3/

[46] Entitled *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, this IBM ebook also explains business drivers for Hadoop / MapReduce open source technologies and for commercial InfoSphere products: http://public.dhe.ibm.com/common/ssi/ecm/en/iml14297usen/IML14297USEN.PDF

4. Directions of travel

Data-centric design: supporting flexible architectures

Developing systems of systems with loose coupling through appropriate APIs is now firmly established as a design paradigm for Internet services, and this development approach is becoming well established for internal services. Recognition that the ability to share data with other systems is not an add-on, but is a key requirement of every system, is the fundamental step required to move toward a pragmatic data-centric architecture. Aligned with the expanding professionalisation of IT management in HE institutions has come an increase in systems analysis and architecture skills, with organisations beginning to manage their data assets at an enterprise level. This trend is likely to continue and accelerate.

Data-centric architecture is a clear and logical approach for many institutions. Applying data-centric principles and practices in order to move toward such an architecture will lead to the development of systems of systems that are easier to maintain, adapt, and extend. The principles of data-centric design should feed into the planning of all enterprise systems, but not as a 'matter of faith': each decision must be made on its merits.

Research data are likely to continue to present specific challenges for enterprise IT provision, with researchers demanding flexible capabilities, sometimes at short notice. Successful institutions are already developing organisational systems to manage these demands, and find an optimum balance of meeting research requirements whilst maintaining enterprise oversight (Hammond, Hawtin, & Davies, 2010).

At present there is confusion between Open Data, Linked Data, and Linked Open Data. It is quite possible to access open data using technologies other than RDF, and it is quite possible to use RDF to support the design of enterprise systems. A common way of using RDF in an enterprise context is as a kind of translation layer: each system is interfaced to RDF, which provides a common and flexible way of describing information. It is clear that Linked Data and RDF provide new opportunities to systems designers, and their usage will increase as the tools making use of them mature. Linked Data is unlikely to be a revolution in systems design, however; it will more often be an addition to or an evolution of other approaches to data representation and interchange.

Within HE, Linked Data has typically been seen as a way of opening data, through data.uni.ac.uk sites. Whereas this is laudable (and has delivered some real benefits), the majority of the pain points for the institutions do not lie with their open data; they are to be found with interfacing finance, HR, payroll, reporting, timetabling, research management, enrolment and estates systems. These are not open systems, but rather are enterprise systems that have been procured and customised and which contain sensitive data. Many vendors of such systems are developing APIs, and it seems likely that integration of such systems will gradually improve, whether through the use of RDF or through a series of *ad hoc* API connections.

Big data and cloud computing: playing a role, in the right places

Alternative database approaches (NoSQL) provide a valuable additional capability to traditional relational databases, and deciding which to use depends on the applications that will be developed on them. Many enterprise tasks are unsuited to NoSQL systems, but many research tasks may be. It seems likely that the current thriving set of NoSQL tools will continue to grow and that new, exciting systems (such as SciDB) will provide valuable tools to HE and FE organisations.

Universities have a wide range of datasets, but in practice these rarely reach a scale at which it is necessary to host using cloud provision. The analytical tasks particularly suited to distributed processing systems such as MapReduce/Hadoop are at present primarily confined to research-computing tasks.

For the broader concepts of a data-centric architecture, moving to “cloud” does not significantly change the data model or architecture of an organisation (although it might change the system and responsibility boundaries). The potential advantages derive from the business model of cloud computing, making new capabilities cheaper and more flexible than would otherwise be the case.

An increasing set of services and platforms are available as ‘cloud’ services. It is already possible to deliver essentially all enterprise services using cloud provision; but whether to do so or not depends on a careful analysis of the costs, benefits and risks. As the maturity of cloud services continues to grow, it is likely that institutions will gradually incorporate some cloud solutions into their institutional provision at the software, platform, and infrastructure levels.

Evaluation of data-centric architecture options

Within HE and FE institutions, an outright and total re-engineering project to migrate existing business systems to a data-centric approach is unlikely to be a realistic aspiration for many organisations. The limited progress to date toward establishing enterprise architectures demonstrates how challenging architectural development is within these types of institution, and a data-centric architecture may be seen as one specific form of enterprise architecture.

Although a complete re-engineering project may be undesirable or impossible, data-centric architectures are somewhat easier to create in an incremental and gradual manner. As they are loosely coupled, they can be assembled over time – but only if the decision has been made to make this change.

Establishing a data-centric architecture can raise some new problems. For example, it is straightforward to obtain a support contract for a monolithic student records system, but when the student records and finance systems are interdependent and it is the emergent properties that provide business value, who can support the overall system? It is likely that this will have to be an in-house capability. The specific risks and disbenefits will depend on the specific situation.

As with all technology strategy choices, decisions about the concept of a data-centric architecture (or any of the specific technologies discussed above) do not come down to a binary choice: organisations should consider them against other options available and in the context of their own strategies, existing technology platforms, experience, available funding and skills base.

In summary, the concept of a data-centric architecture is a clear and logical approach for many institutions. Applying data-centric principles and practices in order to move toward such an architecture will lead to the development of systems of systems that are easier to maintain, adapt, and extend. The principles of data-centric design should feed into the planning of all enterprise systems, but not as a 'matter of faith': each decision must be made on its merits. The specific technologies discussed in this paper provide a useful suite of capabilities, which should be considered as options when building or purchasing systems.

Glossary

Term	Description
API	An Application Programming Interface (API) provides a channel through which software components communicate with each other.
Atomicity	In database systems, atomicity describes a principle whereby (in each atomic transaction) a sequence of database operations is either *all* completed or *none* of these operations is committed (with 'roll back' to an original state). Guarantees of atomicity are strictly maintained in RDBMS environments and more loosely in NoSQL.
AWS	Amazon Web Services (AWS) is a collection of remote computing services based on a cloud computing platform.
BI	Business Intelligence (BI) technologies provide overviews of business operations via reporting, analytics, data mining, etc.
Big data	Big data is a general term referring to large and complex datasets awkward to work with using traditional database management tools. Often this term is discussed in relation to organisations wishing to extract value from very extensive, dynamic, and diverse collections of data (typically in raw, semi-structured or unstructured formats).
Cloud	The "cloud" is a term generally referring to cloud computing infrastructure, over which computing resources (hardware and software) are delivered as remote services. This term derives from the use of a cloud-shaped symbol as an abstraction for a complex infrastructure in system diagrams.
Cloud computing	Cloud computing provides remote computing services delivered over a network (typically the Internet).

Term	Description
Compute clusters	Compute clusters are powerful systems of loosely connected computers designed primarily for conducting complex calculations in parallel.
CouchDB	Apache CouchDB is an open source NoSQL database that uses JSON to store data and JavaScript as its query language, with easy replication as one of its distinguishing features.
Data bus	In computer architecture, a bus is a subsystem that transfers data between components inside a computer or between computers. As explained on the OMG DDS Website, a software data bus is a distributed application composed of data providers and consumers.
DDS	Data Distribution Service (DDS) for Real-Time Systems is an OMG standard aiming to enable scalable, high-performance, interoperable data exchanges between publishers and subscribers.
ERP	Enterprise Resource Planning (ERP) systems are designed to integrate management information across an entire organisation. ERP systems used by academic institutions include Agresso, CampusVue, Jenzebar, and others.
ESB	Enterprise Service Bus (ESB) is a Service-Oriented Architecture (SOA) model supporting interaction and communication across software applications.
FE	Further Education (FE) in the UK is post-compulsory education distinct from that offered by universities (Higher Education). This can range from basic skills training to higher vocational education.
FHE	Further and Higher Education (FHE) is a collective term referring to institutions delivering either FE and HE services.
Hadoop	Apache Hadoop is an open-source, Java-based software framework supporting data-intensive distributed applications. Hadoop enables applications to work with thousands of computationally independent computers and huge amounts (petabytes) of data.

Term	Description
HE	Higher Education (HE) in the UK is a term describing academic work towards a university degree. Within the realm of teaching, it includes both the undergraduate-level and graduate-level (or postgraduate) education.
HEFCE	Higher Education Funding Council for England (HEFCE) is a non-departmental public body in the United Kingdom responsible for the distribution of funding to Universities and Colleges of Higher and Further Education in England since 1992.
HEIs	Higher Education institutions (HEIs) is a frequently used acronym referring to all universities in the UK.
HESA	Higher Education Statistics Agency (HESA) is the official agency for the collection, analysis, and dissemination of quantitative information about Higher Education in the United Kingdom. Its data collection streams include: information about students, courses and qualifications at HEIs; information about HEI staff; finance records of income and expenditure at HEIs, etc.
HTTP	Hypertext Transfer Protocol (HTTP) is the foundation of data communication for the World Wide Web.
IaaS	Infrastructure as a Service (IaaS) is a cloud-based service model for providing access to virtual machines and other resources.
KIS	Key Information Sets (KIS) are collections of comparable information about full- or part-time UK university undergraduate courses, designed to meet the information needs of prospective students. As of September 2012, HEFCE requires KIS datasets to be published on the Unistats Website.
JSON	JavaScript Object Notation (JSON) is a text-based open standard for representing objects (simple data structures and associative arrays), often used as human-readable data interchange in APIs (and elsewhere). Although derived from JavaScript, JSON is language-independent and parsers are available in many coding languages.

Term	Description
Linked Data	Linked Data is a conceptual and technical approach to data storage and access, which is supported by several standards including RDF and SPARQL.
MapReduce	MapReduce is a programming model (implemented by Google) for processing very large datasets, typically used to perform distributed computing on clusters of computers. MapReduce is a key technology in Hadoop's open-source software framework supporting data-intensive distributed applications.
Mashup	A mashup is a Web-based combination of data and functionality from multiple sources to create new services. Although not formally defined by any standard-setting organisation, this term is often used to describe integration of data sources (using open APIs and Web standards) in a way that produces enriched results not originally intended by those producing raw data sources.
MIKE2.0	Method for an Integrated Knowledge Environment (MIKE2.0) is an open-source methodology for Enterprise Information Management. This methodology is part of the overall Open Methodology Framework and includes a (partial) MDM solution offering.
MDM	Master Data Management (MDM) focuses on the (non-transactional) master data held by an organisation, concentrating effort on aggregating, consolidating, quality-assuring, maintaining and distributing key datasets. Tools used to support enterprise MDM include: SAP NetWeaver MDM, Oracle MDM, IBM InfoSphere MDM, Informatica MDM, and many others.
MongoDB	MongoDB (from 'humongous') is an open source NoSQL document-oriented database system. MongoDB stores data as JSON-like documents with dynamic schemas, making the integration of data in certain types of applications easier and faster.

Term	Description
Normalisation	Normalisation is the process of removing redundancy from tables and fields of a relational database. Whereas carefully normalised tables are typical in RDBMS environments, NoSQL reduces normalisation as trade-off for flexibility and performance across large, dynamic, and complex datasets.
NoSQL	NoSQL provides an alternative approach to database design, often more flexible than in a traditional RDBMS environment.
Open data	Open data is a term recently gaining in popularity with the rise of government initiatives such as data.gov and data.gov.uk, designed to make certain datasets (often publicly funded) freely available for everyone to use and republish without restrictions from copyright, patents or other mechanisms of control.
PaaS	Platform as a Service (PaaS) is one model of cloud service, where customer applications are written using libraries and internal systems provided by the service provider.
Petabyte (PB)	Petabyte (PB) is a very large unit of digital information: one quadrillion bytes (1 PB = 1,000,000,000,000,000 B). By comparison, a byte historically encoded a single character of text. Google processes about 24 petabytes of data per day, and the BBC's iPlayer is reported to use 7 petabytes of bandwidth each month.
QAA	Quality Assurance Agency (QAA) is an independent body monitoring the standards for Higher Education qualifications in the United Kingdom.
RDBMS	Relational DataBase Management System (RDBMS) is a data management system that stores data in related tables. For decades most full-scale databases have been based on the RDBMS paradigm, whereas the NoSQL Movement has recently started to move toward more loosely related data models.
RDF	Resource Description Framework (RDF) is the key standard that underpins Linked Data. RDF is based on a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model.

Term	Description
REF	Research Excellence Framework (REF) is a system for assessing the quality of research in UK Higher Education Institutions (HEIs), replacing the Research Assessment Exercise (RAE) by 2014. REF data management requirements are to be published in 2013.
REST	REpresentational State Transfer (REST) is one approach to designing APIs, with loose coupling between different services. REST is less strongly typed than its counterpart, SOAP, and is based on the use of nouns and verbs with emphasis on readability. Unlike SOAP, REST does not require XML parsing and does not require sending of message headers to and from a service provider.
RESTful	RESTful is a term often used to describe systems designed according to REpresentational State Transfer (REST) principles.
ROA	Resource-Oriented Architecture (ROA) is based on guidelines for a RESTful architecture. A key guideline is that Web applications should expose many URIs, one for each Resource (with any resources unambiguously accessible via a unique URI).
SaaS	Software as a Service (SaaS) is a cloud-based service model for providing access to software remotely, where software and associated data are centrally hosted via the 'cloud'. SaaS is typically accessed by users via a Web browser.
Schema	A schema is a description (and definition) of data structure within a database. The schema of a database system explains how data components are organised and how the database will be constructed (divided into database tables, rows, fields, etc).
Sharding	Sharding is a technique used to scale databases by dividing data into segments and hosting each segment (known as a shard) on a separate server. In the sharding technique called 'horizontal partitioning', rows of database tables are held separately on various database servers or physical locations.

Term	Description
SOA	Service-Oriented Architecture (SOA) provides methodologies for developing software in the form of interoperable services with well-defined interfaces to access them. Rather than defining APIs, SOA defines interfaces in terms of protocols and functionality.
SOAP	SOAP (originally defined as Simple Object Access Protocol) is a protocol specification based in XML for exchanging structured information via Web Services. SOAP provides one approach to designing APIs (often contrasted with REST).
SoS	System of Systems (SoS) is an engineering term describing a collection of systems that pool resources and capabilities to create more functionality and performance than simply the sum of constituent parts.
SPARQL	SPARQL Protocol and RDF Query Language (SPARQL) is a recursive acronym, used as the name for the standard query language for databases holding information stored in RDF format.
Trade-off	A trade-off is a compromise in software design or implementation, which seeks to achieve an appropriate balance between two or more competing characteristics.
UCAS	Universities and Colleges Admissions Service (UCAS) is a British service managing undergraduate admissions to UK universities. It gathers, manages, and publishes rich sets of UCAS statistics including tables aggregating data on applications and numbers of accepted applicants over a number of years.
URI	A Uniform Resource Identifier (URI) is a string of characters used to identify a name or a resource over a network (typically the World Wide Web) using specific protocols.
Web 2.0	Web 2.0 is a loosely defined concept focusing on the network as a platform for information and sharing, interoperability and collaboration on the World Wide Web.
XML	eXtensible Markup Language (XML) is a common and flexible data description language based on open standards.

References

Berners-Lee, T., 2006 [updated 2009]. Design Issues in Linked Data. [Online] Available at: http://www.w3.org/DesignIssues/LinkedData.html [Accessed 12 May 2012].

Bilbie, A., 2012. Email to data-ac-uk@jiscmail.ac.uk email list. [Online] Available at: https://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=ind1204&L=DATA-AC-UK&F=&S=&P=5151 [Accessed 12 May 2012].

Clarke, G., 2010. SciDB: Relational daddy answers Google, Hadoop, NoSQL (Interview with Michael Stonebraker). [Online] Available at: http://www.theregister.co.uk/2010/09/13/michael_stonebraker_interview/ [Accessed 13 May 2012].

Dean, J. & Ghemawat, S., 2004. *MapReduce: Simplified Data Processing on Large Clusters*. [Online] Available at:http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//archive/mapreduce-osdi04.pdf [Accessed 13 May 2012].

DeSantis, N., 2012. "Colleges Are Pressured to Open Up Student Data" *Chronicle of Higher Education*. [Online] Available at: https://chronicle.com/article/Unlocking-Student-Data-Could/131551/ [Accessed 7 May 2012].

Ferguson, N., 2011. *Evaluation of ResearchRevealed project*. [Online] Available at: http://researchrevealed.ilrt.bris.ac.uk/files/2012/02/rr-evaluation-report-final.pdf [Accessed 10 May 2012].

Fisher, D. A., 2006. *An Emergent Perspective on Interoperation in Systems of Systems*. [Online] Available at: http://www.sei.cmu.edu/reports/06tr003.pdf [Accessed 1 June 2012].

Guy, M., 2009. *API Good Practice*. [Online] Available at: http://ie-repository.jisc.ac.uk/344/ [Accessed 12 May 2012].

Hammond, M., Hawtin, R. & Davies, C., 2010. *Advanced ICT support for researchers: review of models*. [Online] Available at: http://www.jisc.ac.uk/media/documents/programmes/researchcommunities/Finalreport_SupportResearchers.pdf [Accessed 13 May 2012].

Hammond, M., Hawtin, R., Gillam, L. & Oppenheim, C., 2010. *Cloud computing for research: final report.* [Online] Available at: http://www.jisc.ac.uk/media/documents/programmes/research_infrastructure/cc421d007-1.0%20cloud_computing_for_research_final_report.pdf [Accessed 13 May 2012].

Hawtin, R., Hammond, M., Gillam, L. & Curtis, G., 2012. *Cost analysis of cloud computing for research.* [Online] Available at: http://www.jisc.ac.uk/media/documents/programmes/research_infrastructure/costcloudresearch.pdf [Accessed 13 May 2012].

Hawtin, R., Hammond, M., Miller, P. & Matthews, B., 2011. *Review of the evidence for the value of the 'Linked Data' approach.* [Online] Available at: http://ie-repository.jisc.ac.uk/559/ [Accessed 7 May 2012].

Jacobson, D., Brail, G. & Woods, D., 2012. *APIs: A Strategy Guide.* 1st ed. Sebastopol, CA: O'Reilly.

Joshi, R., 2011. Data-Centric Architecture: A Model for the Era of Big Data. [Online] Available at: http://www.drdobbs.com/architecture-and-design/229301018 [Accessed 2 May 2012].

Loukides, M., 2012. The NoSQL Movement: how to think about choosing a database. [Online] Available at: http://radar.oreilly.com/2012/02/nosql-non-relational-database.html [Accessed 13 May 2012].

Maier, M. W., 1998. "Architecting Principles for System of Systems" *Systems Engineering*, 1 (4), pp. 267-284.

Mell, P. & Grance, T., 2011. *The NIST Definition of Cloud Computing*, Gaithersburg, MD: NIST.

MIKE2.0 Methodology, 2011a. Master Data Management Solution Offering. [Online] Available at: http://mike2.openmethodology.org/wiki/Master_Data_Management_Solution_Offering [Accessed 4 May 2012].

MIKE2.0 Methodology, 2011b. Master Data Management Solution Offering / MDM Challenge. [Online] Available at: http://mike2.openmethodology.org/wiki/Master_Data_Management_Solution_Offering/MDM_Challenge [Accessed 4 May 2012].

Office of the Deputy Under Secretary of Defense for Acquisition and Technology, 2008. *Systems Engineering Guide for Systems of Systems*, Version 1.0. [Online] Available at: http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf [Accessed 3 May 2012].

Oracle, 2011. *Debunking the NoSQL Hype*. [Online] Available at: http://www.scribd.com/doc/67378641/Debunking-Nosql-Twp-399992 [Accessed 13 May 2012].

PriceWaterhouseCoopers, 2010. *Master Data Governance Model and the Mechanism*. [Online] Available at: http://www.pwc.com/us/en/increasing-it-effectiveness/assets/mdm-model-and-mechanism.pdf [Accessed 4 May 2012].

Sage, A. P. & Cuppan, C. D., 2001. "On the Systems Engineering and Management of Systems of Systems and Federations of Systems" *Information, Knowledge, Systems Management*, 2(4), pp. 325-345.

The Object Management Group, 2012. Data Distribution Service Portal. [Online] Available at: http://portals.omg.org/dds/ [Accessed 2 May 2012].

Warden, P., 2011. *Big Data Glossary*. 1st ed. Sebastopol, CA: O'Reilly.

White, C., 2011. *Using Big Data for Smarter Decision Making*. [Online] Available at: ftp://ftp.software.ibm.com/software/data/sw-library/infosphere/analyst-reports/Using-Big-Data-Smarter-Decision-Making.pdf [Accessed 10 April 2012].

Zikopoulus, P. C. et al., 2012. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. New York, NY: McGraw-Hill. [Online] Available at http://public.dhe.ibm.com/common/ssi/ecm/en/iml14297usen/IML14297USEN.PDF [Accessed 6 September 2012]

About JISC Observatory

JISC Observatory provides prioritised information, analysis, and recommendations regarding emerging innovations (technologies, standards) and their usage relevant to Higher and Further Education.

This work aims to ensure that sector institutions can plan interventions in enough time to sustain world-class education and research.

Observatory process

The JISC Observatory evidence-gathering process draws out tacit knowledge and informed experience of those working at the JISC and its Innovation Support Centres (UKOLN and JISC CETIS) as well as throughout a broad range of sector institutions. Through a methodical scanning and sense-making process, this knowledge is made concrete in the form of TechWatch reports, briefings, and other deliverables associated with events.

JISC Observatory uses methods of consultation with others in the sector to inform its work. For objective analysis, it commissions external authors with relevant expertise for its major reports. Working across administrative areas and domains, it produces guidance of relevance to a broad range of roles.

Feedback on this report

JISC Observatory welcomes feedback on this TechWatch report. Please send detailed comments to Observatory Project Manager Thom Bunting (t.bunting@ukoln.ac.uk) or post your feedback on the page where the final version of this report is released on the JISC Observatory Website: http://blog.observatory.jisc.ac.uk/techwatch-reports/data-driven-infrastructure/

JISC Observatory
Observing trends in innovation

JISC cetis UKOLN