Semantic Web 0 (2012) 1
IOS Press

Lifecycle models of data-centric systems and domains

The Abstract Data Lifecycle Model

Editor(s): Werner Kuhn, University of Münster, Germany
Solicited review(s): Tomi Kauppinen, University of Münster, Germany; Todd Pehle, Orbis Technologies, USA

Knud Möller *
DERI, National University of Ireland, Galway
E-mail: knud.moeller@deri.org

Abstract. The Semantic Web, especially in the light of the current focus on its nature as a Web of Data, is a *data-centric* system, and arguably the largest such system in existence. Data is being created, published, exported, imported, used, transformed and re-used, by different parties and for different purposes. Together, these actions form a *lifecycle* of data on the Semantic Web. Understanding this lifecycle will help to better understand the nature of data on the SW, to explain paradigm shifts, to compare the functionality of different platforms, to aid the integration of previously disparate implementation efforts or to position various actors on the SW and relate them to each other. However, while conceptualisations of many aspects of the SW exist, no exhaustive data lifecycle has been proposed.

This paper proposes a data lifecycle model for the Semantic Web by first looking outward, and performing an extensive survey of lifecycle models in other data-centric domains, such as digital libraries, multimedia, eLearning, knowledge and Web content management or ontology development. For each domain, an extensive list of models is taken from the literature, and then described and analysed in terms of its different phases, actor roles and other characteristics. By contrasting and comparing the existing models, a meta vocabulary of lifecycle models for data-centric systems — the *Abstract Data Lifecycle Model*, or ADLM — is developed. In particular, a common set of lifecycle phases, lifecycle features and lifecycle roles is established, as well as additional actor features and generic features of data and metadata. This vocabulary now provides a tool to describe each individual model, relate them to each other, determine similarities and overlaps and eventually establish a new such model for the Semantic Web.

Keywords: data, data-centric, lifecycle, Semantic Web

1. Introduction

The Semantic Web — a web of data rather than documents, where automatic agents can make sense of information, aiding human users and preventing information overload — is still a young phenomenon, and at this point in time we cannot yet say what shape and form exactly it will take in the years to come. A crucial aspect during this development process is a common understanding of the anchor points of what this web of data sets out to be, regardless of specific technologies, languages or systems. Without this understanding, there is a danger that individual efforts will be incompatible, that there is a duplication of efforts or that the effort as a whole will derail. It is therefore necessary to establish a comprehensive conceptual model and architecture of the Semantic Web: *"the architecture of any system is one of the primary aspects to consider during design and implementation thereof, and the [. . .] architecture of the Semantic Web is thus cru-*

* current affiliation:
Kasabi, UK,
E-mail: knud.moeller@kasabi.com

cial to its eventual realisation" [12]. This architecture and model will then provide the required anchor points and allow for a common understanding.

So far, a number of basic foundations have been laid and are quite well understood: (i) A layer cake of languages and technologies has been proposed (e.g., [3]) — and since been changed and extended many times — to implement the vision of the Semantic Web. (ii) The Semantic Web is rooted in the World Wide Web, and so the technical infrastructure [21] of how communication takes place on the latter also applies to the former. (iii) Suggestions have been made to extend the architecture specification for the WWW, in order to incorporate ideas such as self-describing data [30], that underlie the Semantic Web.

While all these components are part of what we can eventually consider a complete conceptual model for the Semantic Web, other aspects have not yet been taken into account. One of the core assumptions about the Semantic Web is that it is a *web of data* — therefore, one of the missing pieces must be to establish what constitutes data on the Semantic Web, what the role is that it plays, and what its features are. Data is a living thing that moves through various stages, such as creation, publishing, use or termination; data is at the centre of the Semantic Web. Therefore, this paper contributes to the overall task of establishing a conceptual model of the Semantic Web by proposing an exhaustive *lifecycle model of data on the Semantic Web*. We are not aware of any definition of lifecycles specific to data — however, a generic definition of the term is *"life cycle, n. [. . .] In extended use: a course or evolution from a beginning, through development and productivity, to decay or ending."* [1].

The lifecycle model is established in three main steps: (i) In light of the fact that the Semantic Web is, at its heart, a data-centric system, we begin in Sect. 2 with an extensive survey of lifecycle models and similar relevant literature from other data-centric domains and systems. (ii) As a second step, in Sect. 3, we then move to a higher level of abstraction by distilling a meta-model and vocabulary of terms (phases, features, roles, etc.) from the survey. We call this model the *Abstract Data Lifecycle Model* (ADLM). In defining the ADLM, we also revisit the surveyed lifecycle models and classify them according to the abstract model. (iii) Finally, in Sect. 4, we turn our attention to the Semantic Web as a concrete example of a data-centric system and apply the ADLM to it, in effect proposing a descriptive lifecycle model of data for the Semantic Web.

2. Data lifecycles in data-centric domains

The idea of a lifecycle for data has been discussed for various different domains in which the generation and use of data and metadata is of central importance. Examples of such data-centric domains are digital libraries, multimedia, eLearning, knowledge and Web content management systems or ontology development. In this section we perform a survey of examples from each domain.

2.1. Lifecycles for multimedia

[15] sets out to find a canonical description of the processes involved in (any) media production. For each of these processes, different tools are available to the user. However, since a unifying model of media production as a whole is not available, the inputs and outputs of the different processes are often not compatible, in particular regarding metadata, thus making it impossible to devise an integrated tool chain. The incentive for proposing a canonical model then is to facilitate the integration of processes and tools. To illustrate the kind of integration they have in mind, the authors make a comparison with UNIX pipes.

The authors define nine different processes, together with the inputs and outputs of each — in fact, the formal definition of inputs and outputs is what clearly sets Hardman's model apart from all other models featured in this paper. Inputs and outputs can be complex objects, and the model is formalised by typing each component and identifying it with an id. The processes are (i) *premeditate* — decisions, inspirations and thought processes that take place prior to the creation of actual media content (e.g., *who*, *where*, *why* or *what*), (ii) *capture* — the process of physically capturing a piece of media content, (iii) *archive* — storing and indexing a media asset, alongside its annotations, (iv) *annotate* — adding arbitrary information to a particular media asset, (v) *query* — retrieving a media asset from an archive, (vi) *message construction* — defining the intended meaning of the media object currently in production, (vii) *organise* — composing a set of media assets to a larger document, (viii) *publish* — converting a document into a format ready for external use, and (ix) *distribute* — making a media document available to external users. An interpretation of the arrangement and interaction of these processes is shown in Fig. 1. The various kinds of `xxxID` in the figure denote named data objects which are part of the input or output of a process, while arrows indicate how

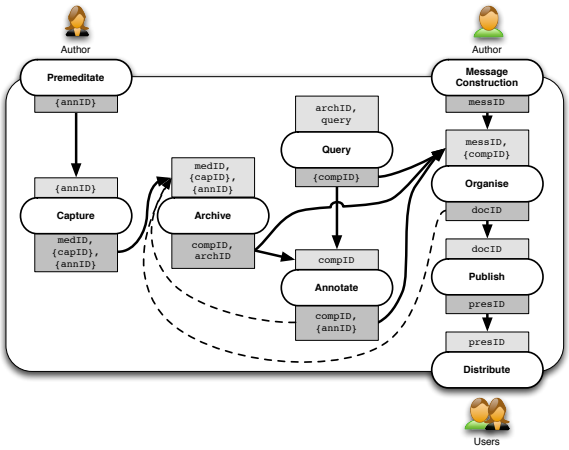

Fig. 1. Processes in media production, from Hardman

the different processes can be linked through their inputs and outputs. Even though this is not conveyed in the figure, some of those data objects can themselves be complex objects. E.g., an annotation referenced by an `annID` will contain a reference to an ontology or vocabulary (`ontID`), as well as a term from that vocabulary (`attID`). Similarly, an archived media asset (`compID`) will contain a reference to the physical media asset (`medID`).

The production model described by Hardman is not necessarily a repetitive life-cycle, but rather a non-circular workflow. The *publish* and *distribute* processes are considered to be the end of the workflow: "Once a document structure is published it is no longer part of the process set." However, the authors also acknowledge that there can be circular sequences: "In a number of cases, the results of a process can feedback into a different process in a different role." An example of this is the idea that an annotation can itself be considered a media asset, which could be archived, annotated, etc. Similarly, a document as the output of the *organise* process can be considered a media asset and as such be archived into the system again. These kinds of transformations are indicated by the dotted arrows in the figure.

[25] discusses the lifecycle of metadata in the context of multimedia systems (here focussing on the multimedia database management system CODAC[1]). Metadata in this context relates to both structure and format of media resources (e.g., resolution, sample rate, color scheme, etc.), as well as their content (what is shown in a video, what can be heard in an audio recording). Kosch characterises those two kinds of metadata as "metadata for content adaption" and "metadata for search and retrieval", respectively (see Sect. 3.5).

The lifecycle model presented by the authors is somewhat unusual, in that it introduces so-called *lifecycle spaces* as one of the central concepts, which is used as a term to denote the different divisions of a multimedia system in which metadata has relevance. Three such spaces are identified: the *content space*, *metadata space* and *user space*. The content space is the main anchor point and is concerned with the multimedia data itself. It comprises a *production/creation*, *postproduction*, *delivery* (e.g., through streaming) and *consumption* stage. The metadata space on the other hand is divided into *metadata production* and *metadata consumption*. Metadata production takes place during content production/creation and post-production, and both phases can produce content-related as well as structural metadata. Metadata consumption takes place during content delivery and content consumption. Finally, within the user space the authors identify *content providers/producers*, *processing users* and *end users*. Both content providers and processing users are involved in metadata production, and both can generate content-related as well as structural metadata. On the other hand, end users are only involved in metadata consumption. This of course also means that the authors do not assume that end users contribute to the media's metadata through means such as tagging. Figure 2 illustrates our own interpretation of how the different spaces are related.

Even though the authors use the term "metadata lifecycle", their model does not contain an immediately apparent circular flow. However, the CODAC system contains router and proxy elements which potentially reproduce metadata used to adapt multimedia data to the preferences or context of a particular user. According to the authors, this in effect closes the metadata lifecycle.

MPEG-7 is adopted as the sole metadata schema for CODAC, apparently with the implicit assumption that it suffices for the purpose of multimedia data. For this reason, the question of ontology creation or validation of a chosen terminology is not considered.

[1] `http://www-itec.uni-klu.ac.at/~harald/codac/`, retrieved 31/10/2010

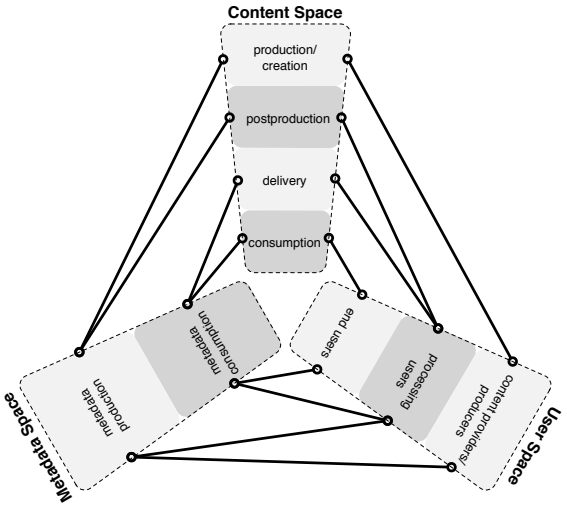

Fig. 2. Metadata lifecycle spaces in multimedia, from Kosch et al.

2.2. *Lifecycles in eLearning*

[5] investigates the lifecycle of data in the context of eLearning. Their approach is to first give an overview of a number of other lifecycle models for learning objects (LO), from which they then distil their own common terminology and model. The end-result is an elaborate model describing a combined lifecycle of the actual LOs and their associated metadata.

The model consists of nine different phases, which are related to the lifecycle stages of the reviewed models. Those terms are: *initiation*, *conception*, *realisation*, *classification*, *validation*, *diffusion*, *usage*, *feedback* and *termination*. Strictly speaking, *initiation* takes place before any actual work on the LO begins, and is therefore outside of the cycle. Only in *conception* does the work begin in earnest. Various data elements are defined, as well as their relations to other LOs. *Realisation* marks the point when the actual content of the LO is produced, while *classification* positions it in the context of a classification system such as the Dewey Decimal system. *Validation* is defined as a means of quality control by experts, which may lead to the LO being pushed back to the classification, realisation or even conception step. In *diffusion*, the LO is being introduced into a concrete learning management system, where it becomes available to learners in the *usage* step. User reactions and comments are analysed during *feedback*, which may again lead to changes of the LO in terms of conception or realisation, or even to the complete *termination*, in case it is deemed unsuitable or unsuccessful.

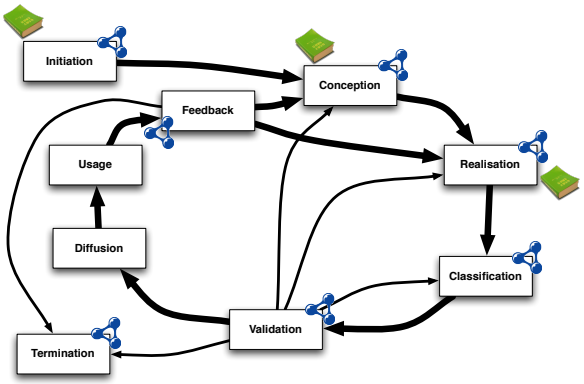

Fig. 3. Metadata lifecycles for learning objects, from Catteau et al.

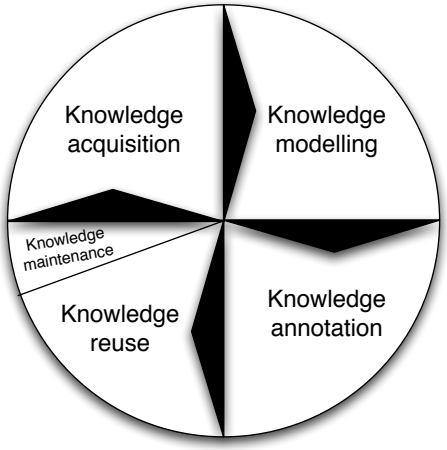

Fig. 4. Metadata lifecycles for learning objects, from Millard et al.

In Fig. 3, the thick arrows indicate the main flow of the proposed lifecycle model, while thin lines indicate changes after "unfavourable evaluation" in the validation or feedback stage. Furthermore, the authors identify three stages which involve changes to the LOs themselves (indicated by a book icon), while there are seven different stages which involve changes in the LO metadata (indicated by an RDF icon).

Another, much simpler model is introduced in [32]. The goal of their "Knowledge Life Cycle" is to facilitate the lifting of learning objects and their metadata to a level of formal semantics. They propose four basic stages: *Knowledge acquisition*, which is the conceptualisation of a knowledge domain with the help of domain experts, *knowledge modelling*, which is the formalisation of the conceptualisation into an ontology language, *knowledge annotation*, which is the an-

notation of learning objects with the ontology and finally *knowledge reuse*, which is usage of the annotated LOs in applications such as search, automatic composition of courses, personalisation, etc. Millard's lifecycle model is circular in the sense that one can have several iterations, with an additional *knowledge maintenance* step in between reuse and acquisition (see Fig. 4). Apart from its simplicity, a crucial difference between Catteau's model and Millard's is that the former completely omits ontology creation, whereas the latter gives it a very prominent position (by taking up half of the cycle).

2.3. Lifecycles in digital libraries

Another domain for which data lifecycles are relevant are digital libraries. [6] describes a very detailed and elaborate 10-step methodology for setting up metadata systems for digital libraries (or more generically: collections of digital artefacts). The lifecycle model is divided into four general groups: *Requirement Assessment and Content Analysis*, *System Requirement Specification*, *Metadata System* and *Service and Evaluation*. Each of those groups is again divided into a number of different steps, as shown in Fig. 5.

The *Requirement Assessment and Content Analysis* phase covers aspects such as defining the basic metadata needs of the task at hand (e.g., schedule, scope or function), as well as deep needs (ontologising the domain, defining search strategies, etc.) and a review of candidate metadata standards. Since the authors assume a situation where data from legacy systems will be migrated into the new system, another aspect of this phase is the analysis of the legacy system in terms of access options, suitability for transformation, etc. After this, the *System Requirement Specification* phase will produce a detailed specification document (the *Metadata Requirement Specification*, or MRS), covering all aspects which were dealt with in the assessment phase. Also covered by this phase is an evaluation of both existing metadata platforms (software) for use in the desired target system and the possibility to develop such a system from scratch. In the *Metadata System* phase, two things take place: a best practice guideline for applying the MRS to specific metadata elements is prepared, and the development or setting up of the system which was decided upon in the previous step is being done. Finally, in the *Service and Evaluation* phase, a metadata service model (including a service mechanism, different user roles and their relationships) is put into place to ensure the smooth running of the system. Also, an evaluation of the system takes place, which can feed back into a new round of the cycle.

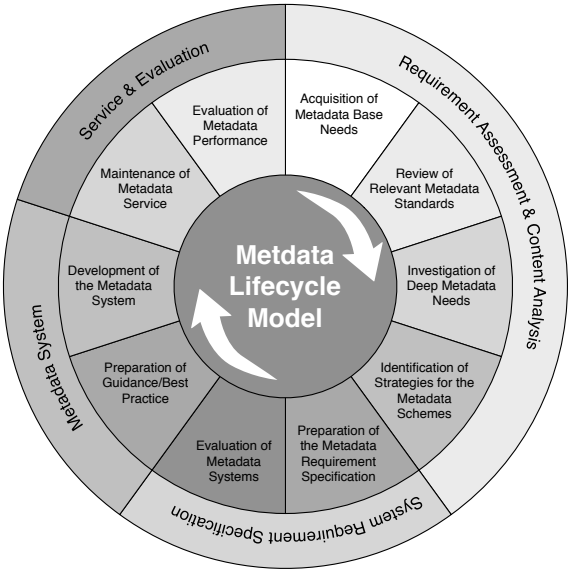

Fig. 5. Metadata lifecycle for digital libraries, from Chen et al.

If we compare Chen's model to the other models discussed in the paper, it becomes obvious that it isn't really a lifecycle model for data, but rather a lifecycle model for ontology (see Sect. 2.5) and systems development. All data and metadata in the system comes from translating resources from legacy systems into a format defined by the new ontology. The creation of new instance data does not feature anywhere in the model.

In [39], a lifecycle model for ontology development which was previously defined in [38] is applied to the domain of digital libraries. Also this model focusses on the ontology development aspect of digital libraries. While much simpler than [6] (including *creation*, *evaluation*, *negotiation* and *versioning* phases), it also specifically distinguishes between automatic creation of ontologies (*ontology learning*) through machines and manual, collaborative ontology creation of ontologies through humans (see Sect. 3.4).

2.4. Lifecycles for knowledge and content management

Knowledge and content management represents the largest subset of data lifecycles discussed in this paper. The domain is defined very loosely, comprising both

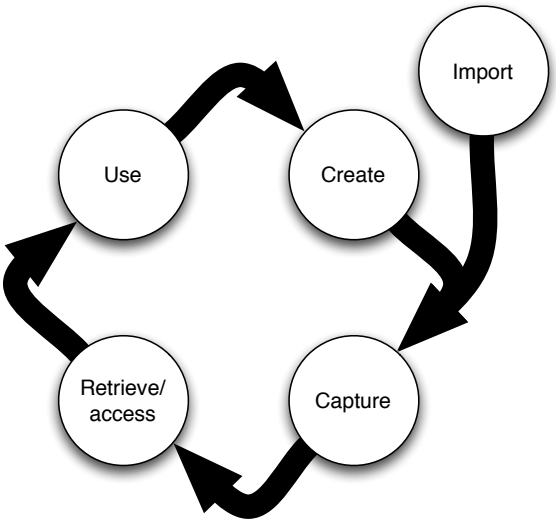

Fig. 6. Knowledge process, from Staab et al.

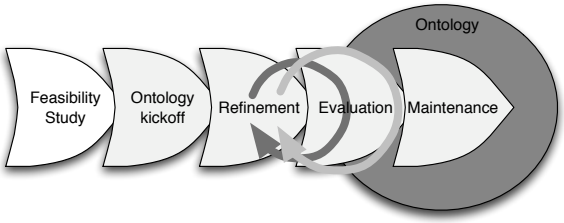

Fig. 7. Knowdlege metaprocess, from Staab et al.

traditional knowledge management and lifecycles with a specific Web or even Semantic Web focus.

[42] proposes a metadata lifecycle for knowledge management (KM) in organisations. In fact, the authors define two such cycles, which are intertwined: the *knowledge process* and the orthogonal *knowledge metaprocess*. The former denotes creation, retrieval and use of actual knowledge (not simply in the form of documents, but as *knowledge items*). On the other hand, the knowledge metaprocess denotes the process of planning and implementing an infrastructure and platform for knowledge management, including ontology creation and management. For both cycles, the authors assume that a soft- and hardware infrastructure is already in place.

Within the knowledge process (i.e., the lifecycle of data), the authors identify four different steps: *Create*, *Capture*, *Retrieve/Access* and *Use*, as depicted in Fig. 6. In addition, *Import* is identified as a fifth step, which allows to introduce new, external data (e.g. from documents). Both Creation and Import are characterised by the fact that new knowledge items come into the system. Knowledge items have varying degrees of formality (from free text documents over templated documents to formal knowledge structures), and can thus be said to cover both data and metadata in the classical sense. In the Capture phase, the new knowledge items are then integrated into the system: this can mean indexing, additional annotation, establishing relations to other knowledge items, etc. Retrieve and Access usually involves querying and browsing the organisation's knowledge base. In other words, this phase defines a typical search task of a knowledge worker, at the end of which they have retrieved a number of knowledge items. The authors stress that, once knowledge items are found, the job is not done: instead, another important phase is Use, which can span aspects such as personalisation, pro-active access, integration with other applications or aggregation of separate knowledge items into a new whole. Aggregation and inference can lead to new knowledge and thus close the cycle.

The knowledge metaprocess is mostly concerned with ontology development for a particular organisation or use case. It consists of a *Feasibility Study*, *Ontology kickoff*, *Refinement*, *Evaluation* and *Maintenance*. The Feasibility Study takes place before the actual work on the ontology begins. Its purpose is to properly specify the use case at hand, define the organisational context and identify any problems which might appear along the way. The purpose of the kickoff phase is to produce a requirements document, which will serve as a guide throughout the project, elaborating on aspects such as goal, domain and scope, supported applications, available knowledge sources, users and external ontologies which could potentially be reused. In the Refinement phase, the ontology engineers will then start to implement the specification. The suggested approach is to start with a simple baseline taxonomy, over a so-called "seed ontology" until one arrives at the final target ontology, which is expressed in a formal representation language. For the Evaluation phase, the designers refer back to the specification document to check their ontology, test it within the targeted applications and gather feedback from beta users. Finally, the Maintenance phase ensures that the ontology always stays up-to-date and reflects changes in the real world. Both in the Evaluation and Maintenance phases, user input and feedback may lead to further refinement, as indicated by the loops in Fig. 7.

[14] introduces a framework for the description of generic metadata generation. She focusses on the different *processes*, *people* (roles) and *tools* involved. While not strictly speaking a complete lifecycle model, her framework is still providing important input for at least part of such a model. In terms of processes, Greenberg establishes the very broad dichotomy of *human metadata generation* vs. *automatic metadata generation*. While the first was historically the only form of metadata generation, the latter is becoming increasingly important, especially considering the huge amounts of data on the Web and elsewhere today. In *human metadata generation*, the different classes of persons which are proposed in the framework are *professional metadata creators* (e.g. professional librarians), *technical metadata creators* (e.g. library assistants), *content creators* (e.g. authors) and *community or subject enthusiasts*[2] (e.g. readers). While those classes can be arranged on a vector of decreasing professionality, they are not necessarily disjunct. In terms of tools, the framework distinguishes between three different kinds: *human beings*, *standards & documentation* and *devices*. The latter are the technical means to actually capture metadata, and are further divided into *templates*, *editors* and *generators*. *Templates* are defined as simple forms to input metadata, while *editors* will aid the user with links to documentation, syntactical help, etc. *Generators*, finally, will produce metadata automatically and can often be found as components in editors.

[27] proposes a simple lifecycle model for knowledge management in organisations. The model comprises four rather high-level steps, which are discussed by suggesting the different information technologies (IT) which facilitate them: (i) *Knowledge capture* is defined as "the process by which knowledge is obtained and stored". Facilitating ITs are database systems, data warehouses and document management systems. (ii) *Knowledge development* is the organisation and analysis of data "for strategic or tactical decision making". Examples of ITs used in this step are data mining tools, OLAP (online analytical process) and competitive intelligence systems. (iii) *Knowledge sharing* means distribution of knowledge and is enabled by group support systems and communications technology such as EDI (electronic data interchange), e-mail, voice mail, video conferencing and electronic bulletin boards. (iv) *Knowledge utilisation* finally is the use "without computer knowledge" of knowledge by end users in the organisation. ITs used in this step are vaguely defined as GUI-driven end user applications. Multimedia is also mentioned as an enabler technology.

In a general paper on Web content management (WCM) systems in organisations, [29] suggests a lifecycle model for WCM as a special case of KM. In her model, *collection* and *delivery* are two processes which are repeated iteratively for as long as the organisation is involved in WCM. Collection is the authoring or creation of content, while delivery means the publishing or deployment of data. In addition to those two processes, the *workflow* and *control and administration* processes have a ongoing supportive function. The former enables collaboration and manages steps such as approval, development, etc., while the latter includes the identification of user roles, groups, management of the system, security and similar aspects.

On the fringes of knowledge management, [19] proposes a lifecycle model for digital curation, to be promoted by the UK Digital Curation Centre (DCC)[3]. It is one of the most complex models in this survey, distinguishing between three different types of lifecycle actions, from *full lifecycle* actions, which take place continually, over *sequential* actions, which conform more to the traditional lifecycle concept and form the bulk of the model, to *occasional* actions, which only take place potentially at certain moments in the lifecycle. Full lifecycle actions are mainly administrative and preparatory, while the latter two kinds of actions deal with the data directly. As full lifecycle actions, the DCC defines *description and representation information*, similar to ontology development phases of other models, *preservation planning*, which provides planning for the management and administration of the regular lifecycle actions, *community watch and participation*, which ensures involvement in related activities (e.g., standards development), and finally *curate and preserve*, which drives the actual management and administration of the lifecycle. Sequential actions are *conceptualise*, preceding the creation of data, *create and receive*, either creating new data or importing existing data, *appraise and select*, evaluating and selecting data for long-term curation, *ingest*, archiving data in a repository, *preservation action*, which includes

[2] Greenberg relates an interesting account of a project at the Fine Arts Museum of San Francisco, which involved voluntary community enthusiasts to help assigning keywords to a corpus of 20.000 images — an early, pre-Web2.0 example of collaborative tagging.

[3] `http://www.dcc.ac.uk/`, retrieved 31/10/2010

data cleaning and validation or assigning preservation metadata, *store*, another form of archiving, *access, use and reuse*, making data accessible to potential users, and finally *transform*, meaning to create new versions of data already present in the lifecycle. The three occasional actions are *dispose*, which can take place after *appraise and select* in order to remove data from the lifecycle, *reappraise*, which is a loop back from *preservation action* to *appraise and select*, and finally *migrate*, which moves directly from *preservation action* to *transform*, skipping the intermediate actions.

In [33], a very simple lifecycle called MAAME is proposed for semantic applications. The name is derived from the five phases which constitute the lifecycle: *modelling*, *application*, *authoring*, *mining* and *evaluation*. The *modelling* phase covers all elements of ontology modelling, whereas *application* denotes the process of defining how semantic information will be used in a concrete application, i.e., what is the role of semantics in that application. In *authoring*, instance data is created manually, while *mining* creates data automatically from other sources, through techniques such as natural language processing or machine learning. Finally, *evaluation* encompasses all aspects of feedback about the lifecycle, including the model, the application itself and its instance data. Mödritscher discusses the lifecycle itself only at a very high and abstract level, but then grounds it by mapping four different concrete applications to it.

[17] loosely defines a lifecycle for linked data [2], with the express purpose of categorising different R&D projects in a research group. In this sense, the six phases of *data awareness*, *modelling*, *publishing*, *discovery*, *integration* and *use cases* are used. The first of these phases should probably be seen separately, in that it does not concern a particular unit of data or a particular dataset, but rather the awareness towards linked and open data in general.

2.5. Ontology lifecycles

A special case of data lifecycles are ontology lifecycles, i.e., lifecycle models that describe aspects such as creation, integration or reuse of ontologies (as opposed to instance data). In fact, some of the metadata lifecycles presented in the previous sections, e.g. Chen et al., turn out to be ontology lifecycles, while others at least include the concept of ontology creation prominently (Staab et al., Millard et al., Mödritscher, Hausenblas). A significant number of what can be considered lifecycle models for ontologies have been published under the label of *methodologies* for ontology development and often do not use the term "lifecycle" at all.

[10] gives a comprehensive overview of different ontology development methodologies based on their compliance with the IEEE Standard for Developing Software Life Cycle Processes (1074-1995) [41]. The methodologies are categorised as (i) *building ontologies from scratch*, (ii) *re-engineering ontologies* or (iii) *collaborative methodologies*. For (i), the authors include the Cyc methodology, Uschold and King, Grüninger and Fox, KACTUS, their own METHONTOLOGY [8,13] and the SENSUS-based methodology. For (ii), they outline an approach of their own, while for (iii), the CO4 and KA^2 methodologies are included. Each methodology is evaluated in terms of whether or not it implements the different processes defined by the IEEE standard. Depending on how many processes are considered by a given methodology, it is given a maturity rating (the more processes, the more mature). The authors' own METHONTOLOGY comes out as the "most mature approach". Additionally, each approach is characterised according to the kind of lifecycle model it proposes (sequential, incremental or evolving), as presented in [8] (see also Sect. 3.1). [9] discusses how the (development) lifecycles of different ontologies can touch due to integration and reuse. Considering the *"confluences and forking of life cycles"*, the authors present a method to make the relations between ontologies explicit and define the steps necessary to integrate them. Other, more recent examples of ontology development lifecycles include work done in the Knowledge Web project[4] (e.g., [38]) and the NeOn project[5].

2.6. Lifecycles in databases

Databases are not an application domain such as the ones presented in the previous sections, but rather an enabling technology. For this reason, they are only briefly mentioned here. Nevertheless, databases are an area where data lifecycles are of central importance, and the CRUD operations [23] in particular are relevant. The CRUD acronym denotes the four fundamental atomic operations common to persistent database systems, i.e., (i) *create*, (ii) *read*, (iii) *update* and (iv) *delete*. The point of view of CRUD is that these are all individual, low-level operations, rather than phases

[4] `http://knowledgeweb.semanticweb.org/o2i/`, retrieved 31/10/2010

[5] `http://www.neon-project.org/`, retrieved 31/10/2010

in a sequential lifecycle. However, there is still an implicit temporal element to them, in that each unit of data first needs to be created before it can be read, updated or deleted. Also, all four operations can be easily mapped to the lifecycle model proposed in the following Sect. 3. Indeed, it is easy to argue that CRUD in fact does represent a simple lifecycle for databases.

While CRUD originates from databases, where it is often used at the basic record level, the term is now also widely used to describe the functionality of applications at a higher, even user interface level. In this interpretation, an application dealing with data entities of a particular type is only considered complete if its UI at least supports the four CRUD operations.

3. The Abstract Data Lifecycle Model

Looking at the survey of lifecycle models presented in Sect. 2, we can see that a significant number of such models have been proposed for different domains. Some models have been designed from scratch, while others have been based on surveys of previous models (in particular McKeever and Catteau et al.). However, none of the literature presented has looked beyond its respective domain and proposed a *generic* lifecycle model for (meta)data. On the one hand, this is understandable, considering that the individual research communities are often disjoint. On the other hand, even though it is not always possible to perform a direct one-to-one mapping between the different models, there are striking similarities among all of them. Each model has its own specific focus, and while each domain is concerned with a different *kind* of data, they all deal with the same thing: data. For this reason, it is feasible to devise a domain-agnostic (meta)data lifecycle model based on the previous section, which will help to compare and point out differences and similarities between different systems with respect to their data lifecycles. In this approach, the abstraction is derived from specific instances. The alternative approach, as e.g. adopted in [26] for the domain of case base maintenance, would be to begin with the abstraction and then use it to classify a selection of instances in the pertinent domain[6].

[6]The bottom-up approach chosen in this paper is not in general better or worse than the top-down approach. However, it ensures a broader coverage and wider applicability, while the top-down approach may be more suitable for a restricted, specific use case.

The meta-model, which we call the Abstract Data Lifecycle Model (ADLM), consists of five parts: (i) a set of *phases* which will generalise the steps and processes defined in the individual models in the survey (Fig. 8), (ii) a set of *features* which can be used to define lifecycle models further (overview in Tab. 2), (iii) a set of *roles* describing the different actors in the model (Fig. 8), (iv) a set of features describing the *actors* in the model and (v) a set of features describing the *data and metadata* found in a lifecycle. Finally, each of the lifecycle models in the survey will be categorised in terms of the meta-model.

3.1. Lifecycle phases

The choice of phases for the meta-model is based on the survey in Sect. 2. While some of the reviewed models go into a lot of detail and include domain-specific aspects, or aspects which pertain to specific implementation matters, other models are very high-level and generic. Both extremes are undesirable for the purpose of building a comprehensive meta-model, in the sense that they are either over- or under-specific. However, it is still possible to map all models covered in the survey to the meta-model.

Like the models covered in the survey, the ADLM has transitions between the different phases. In many cases, these transitions are possible in both directions, allowing to pass the phases in many different orders, including recursively. While there is no explicit *versioning* phase or aspect to the ADLM, one implication of the recursiveness is that versioning of data is possible within the ADLM.

Ontology development In this phase the formal domain model for (meta)data is defined. As the previous sections show, the literature contains several lifecycle models which are dedicated solely to the domain of ontology development (notably Chen et al. and the methodologies in Sect. 2.5), whereas others do not consider ontology development at all. There are a few examples in which ontology development is seen as a component in a larger structure, such as Millard et al., in which *knowledge acquisition* and *knowledge modelling* are the two initial steps in the five-step cycle, or Staab et al., where the *knowledge meta-process* is one of the two cycles which make up the model as a whole. Nevertheless, the reason why ontology development is not featured at all in some of the literature is not that it is considered unimportant, but simply that it is considered to be something that has taken place

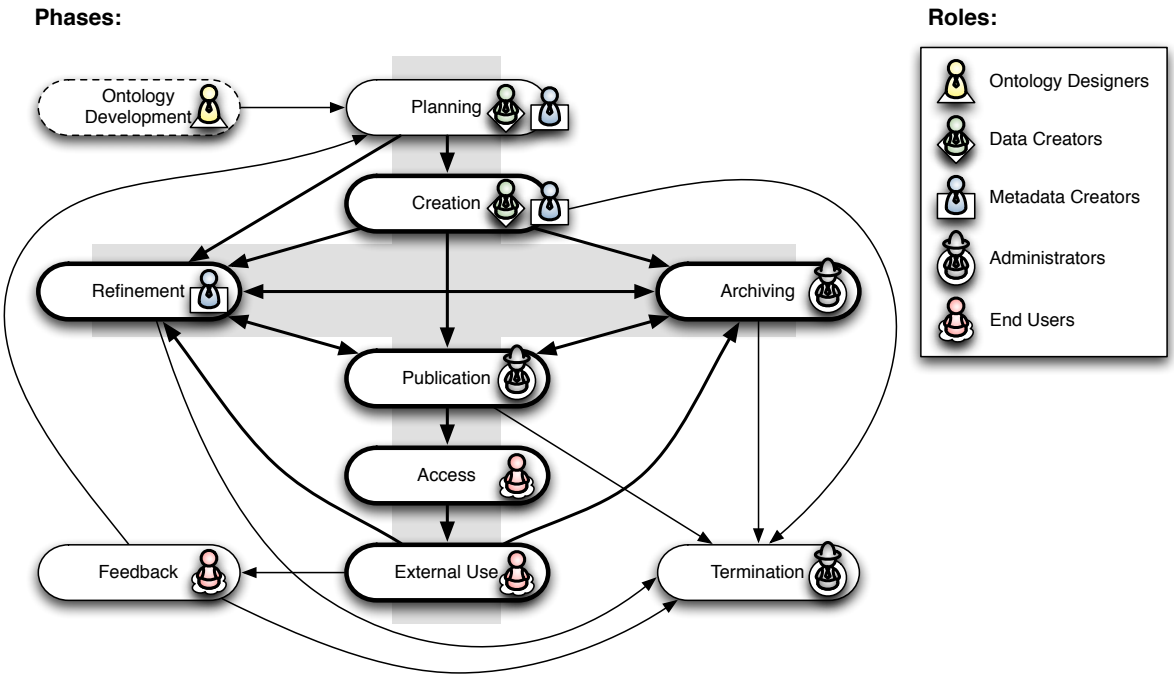

Fig. 8. Phases and roles of the Abstract Data Lifecycle Model (ADLM)

a priori to the lifecycle of instance data. E.g., Kosch et al. simply assume that MPEG-7 will be used, while Catteau et al. suggest to use the Dewey Decimal (DD) or a similar system. However, both MPEG-7 and the DD have obviously been designed at some stage and are in some form of lifecycle themselves. While it may be less important in some domains, ontology development is conceptually still a pre-requisite to any data lifecycle in a data-centric system, which is why it is included in the ADLM. It is, however, placed outside the core part of the lifecycle. It can also be considered a lifecycle of its own, but one which must precede every other (meta)data lifecycle at some point.

If, in a particular lifecycle model, ontology development is included as a phase, this inevitably means that the lifecycle is not that of a particular piece of data or metadata, but instead of a complete system as a whole, which includes both ontology and instance data. This characteristic sets the ontology development phase apart from all other phases, which is reflected in Fig. 8 by using a dotted line for its outline.

Planning *Planning* is a phase which precedes the actual lifecycle of data and describes the moment when the intent to create data takes a concrete form, but before the data is created as part of the system. As Hardman puts it, planning is "outwith the system", but it can already result in a number of facts which later may become part of the data, e.g. ownership, intent, etc. It is obvious that some form of planning must always take place, but only a few models in the literature name it explicitly. In Catteau et al. this phase is called *initiation*, while Hardman has two different processes which fall under planning: one is called *premeditate*, and denotes the planning of creating a single unit of data, while *message construction* explicitly means the planning and intent of combining several pieces of data to a larger whole. From the point of view of the ADLM, both processes shall be considered the same (but possibly during different iterations).

Creation The *Creation* phase defines the moment when new data or metadata is created in terms of the system at hand. This data is either genuinely new and has not existed in any other (formal) format before, or it has been imported from another system — the main point is that it did not exist *within the lifecycle's system* before the creation phase. All models discussed above contain a creation step: the multimedia model in Kosch et al. calls it *Production/Creation*, while Hardman names it *capture* (here in the sense of "capturing a piece of media with a recording device").

Within eLearning, Millard et al. include a *knowledge annotation* step (which in fact spans the phases of data *creation*, *archiving* and data *refinement*, see below), whereas Catteau et al. split creation into the two separate *conception* and *realisation* steps. In knowledge management, Lee and Hong call this phase *knowledge capture*, McKeever *collection* and Staab et al. have *create*. The latter also distinguish explicitly between the creation of new data and the *import* of existing, but external data, i.e., the transformation of existing data from one format into the format and infrastructure of the lifecycle's system. Finally, this phase in combination with the following *archiving* phase is represented by the *create* operation in CRUD set of operations, since it also entails making the newly created data persistent in the system.

Archiving *Archiving* denotes the process of anchoring a piece of data within the system by the means of indexing, cataloguing or a similar activity. Not all models covered in the survey feature anything that can be mapped to this phase. However, Hardman introduces an *archive* step, whereas Staab et al. suggests a *capture* step, which combines aspects of the *archiving*, *refinement* and *publication* phases (see below). In McKeever, archiving is covered by the supporting process *control and administration*.

Refinement The *refinement* phase covers all kinds of activities which make additions or changes to data that already exists within the system. In a very general sense, this can mean to *annotate* data, which incidentally is the name of one of the steps within the refinement phase in Hardman, the other one being *organise*, which means the combination of several data items to a larger whole. Of course, depending on whether data and metadata are distinguished, annotation as refinement could also be interpreted as another iteration of the *creation* phase: if all data in the lifecycle is considered equal, then adding new annotations to a piece of data is nothing else than creating new data. This is reflected in Hardman by the fact that the output of the *annotate* process can be new data that can enter the system. In Kosch et al., refinement is named *post-production*, a term typically used in the media domain, for which this model was made. Catteau et al. emphasise the act of *classification* as a specific kind of refinement. Both post-production and classification can be considered specialised, restricted kinds of annotation, i.e., refinement. On the other hand, Lee and Hong introduce the very generic *knowledge development* process, which denotes various kinds of data analysis activities. The *knowledge annotation* and *capture* steps in Millard et al. and Staab et al., respectively, both contain aspects of refinement, but are not limited to this phase. Like archiving, McKeever covers refinement as part of the *control and administration* process. In Greenberg, refinement is reflected as the general process of *metadata generation* and further distinguished into *human metadata generation* and *automatic metadata generation*[7]. In CRUD, the *update* operation entails a combination of *refinement* and *archiving*, since any changes made to data are immediately made persistent in the system.

Publication The *publication* phase is the moment when data is made accessible to the users either within or outside of the system, or both. Most of the lifecycle models in the survey have a dedicated slot for publication: For multimedia data in Kosch et al. this is called *delivery*, while Hardman splits this phase up into a *publish* (which really is just preparing a piece of data for publication) and a *distribute* (the actual act of publication) step. In eLearning, LOs are being made available to the system in the *diffusion* step. In Staab et al., publication is part of the very broad *capture* step, while Lee and Hong reserves a dedicated slot with *knowledge sharing* and McKeever with the *delivery* step.

Access *Access* denotes the moment in the lifecycle when parties from either within or outside of the system gain access to the data in the system, e.g., by means of a query or through browsing. This phase is covered explicitly in Hardman by the *query* step, and in Staab et al. by the *retrieve/access* step. Kosch et al. and Catteau et al. also recognise access as a necessary step in the lifecycle, as *consumption* and *usage*, respectively, but do not distinguish it from actual use (see below). The *read* operation of CRUD is also an instance of access in terms of the ADLM.

External Use Whereas *access* merely means to retrieve data, *external use* implies that the user then performs some further actions with it, such as export into other systems or software or aggregation. It should be stressed that this phase explicitly means usage of the system's data *outside* of it. If data is used and changed within the system, this is a case of *refinement*. As mentioned above, use falls together with access in both Kosch et al. and Catteau et al., whereas Staab et al. include a dedicated *use* step. Also, the model in Millard

[7] This distinction is extended to a general feature of lifecycle actors in the model proposed here; see Sect. 3.4.

et al. includes a dedicated *knowledge reuse* step, and Lee and Hong suggest the term *knowledge utilisation*.

Feedback The *feedback* phase allows users of a system to comment on the data or metadata they have previously accessed and used. In a strict view, this only makes sense in centralised lifecycles, where there is some sort of authority the users can give feedback *to*. Nevertheless, the multimedia model in Kosch et al. does not provide for any kind of explicit feedback. In Catteau et al., different kinds of users (experts and end users) provide feedback at different levels in the *feedback* and *validation* steps. Staab et al. also provide feedback as *evaluation*, but only for the ontology which was devised in the knowledge metaprocess. In the same sense, Chen et al. envisages a *Service & Evaluation* phase, which provides feedback on the ontology and overall performance of the system. Similarly, almost all dedicated ontology development methodologies provide the opportunity for feedback, usually in order to gain consensus within a community. The *knowledge maintenance* phase in Millard et al. has a similar purpose.

Termination Finally, the *termination* phase presents the moment when data is removed from the system. In other words, it is the "end of life" phase (a term taken from product lifecycles). Surprisingly, of all the models in the survey, only Catteau et al. provide a slot for the termination of data. Termination is, however, one of the four CRUD operations, i.e., the *delete* operation.

3.2. Lifecycle features

There are a number of dichotomies and other characteristics that can aid in classifying different lifecycle models and which are useful in pointing out the differences and similarities between the various lifecycle models presented in Sect. 2.

Distinction data vs. metadata An important point in looking at data lifecycles is the question whether or not a distinction between data and metadata is made. Are both kinds of data first-class citizens? A number of models in the survey clearly make this difference: in multimedia, Kosch et al. distinguishes between multimedia resources and metadata about those. In eLearning, both Catteau et al. and Millard et al. consider learning objects and annotations on them as different elements in their lifecycle models. While McKeever doesn't discuss metadata in Web content management in detail, she does seem to distinguish the two when she states that part of the *control and administration* process is the "ability to specify metadata". On the other hand, Hardman initially distinguishes between data (media objects) and metadata (annotations on media objects), but leaves the possibility open that annotations themselves can become first-class citizens (see Sect. 2.1). Also in the model in Staab et al., the distinction is not so clear. Instead, all data in this model are knowledge items with a varying degree of formality, ranging from unstructured text over templated data to structured data objects. Finally, Lee and Hong only use the generic term "knowledge" when describing the data in their lifecycle. Since activities such as data mining can lead to new knowledge, we will assume that a clear distinction between data and metadata is not made in their model.

Prescriptive vs. descriptive The dichotomy of *prescriptive vs. descriptive* is typically used in linguistics, where it denotes the two different approaches to language in general, and grammar in particular. *Prescriptive* linguistics attempt to define a set of rules for the "correct" use of language, whereas *descriptive* linguistics try to analyse a language in its actual use, in order to find rules and regularities within it (see e.g., [7]). Applied to the domain at hand, we use the term *prescriptive* for lifecycle models which try to establish a set of steps which are then suggested for use by others[8], while a *descriptive* model will look at a given system and find a lifecycle in it. Using this terminology, many of the models presented in Sect. 2 can be called prescriptive, because they suggest to the reader a methodology of how the lifecycle of data or metadata should be handled. Hardman presents an interesting exception to this trend: "The processes should not be viewed as prepackaged, ready to be implemented by a programmer. Our goal is rather to analyse existing systems to identify functionality they provide". Lee and Hong and McKeever both address an audience of decision makers or experts in a business position, rather than one of implementers and developers. For this reason, they simply report what they conceive to be lifecycle processes in knowledge and Web content management, rather than suggest a particular model to use, and should therefore be considered descriptive.

[8] In a very combative position paper, [28] make the point that, applied to the area of software development, prescriptive lifecycle models may in fact be considered a bad idea in many circumstances.

Table 1
Comparison of Metadata Lifecycles — Phases

Meta-Model	**Kosch** (Multimedia)	**Hardman** (Multimedia)	**Catteau** (eLearning)	**Millard** (eLearning)	**Staab** (KM)	**Lee** (KM)	**McKeever** (WCM)	**CRUD**
Ontology Development	—	—	—	*knowledge acquisition, knowledge modelling*	(*knowledge meta-process*)	—	—	—
Planning	—	*premeditate*	*initiation*	—	—	—	—	—
Creation	*production/ creation*	*capture*	*conception, realisation*	*knowledge annotation*	*create, import*	*knowledge capture*	*collection*	*create*
Archiving	—	*archive*	—	*knowledge annotation*	*capture*	—	*control and administration*	*create* and *update*
Refinement	*post-production*	*annotate, organise*	*classification*	*knowledge annotation*	*capture*	*knowledge development*	*control and administration*	*update*
Publication	*delivery*	*publish, distribute*	*diffusion*	—	*capture*	*knowledge sharing*	*delivery*	—
Access	*consumption*	*query*	*usage*	—	*retrieve/access*	—	—	*read*
External Use	*consumption*	—	*usage*	*knowledge reuse*	*use*	*knowledge utilisation*	—	—
Feedback	—	—	*feedback, validation*	*knowledge maintenance*	(*evaluation*)	—	—	—
Termination	—	—	*termination*	—	—	—	—	*delete*

Homogeneous vs. heterogeneous We will say that a lifecycle is *homogeneous* when the data in the system it describes is homogeneous, i.e., when its schema or ontology is known beforehand, and no data of unknown form or using unknown vocabulary terms will typically enter the cycle. In contrast, a lifecycle is *heterogeneous* when the data in the system it describes is heterogeneous, i.e., when its form or ontology is not known beforehand. Most lifecycle models presented in the survey are examples of homogeneous lifecycles, since they only deal with a specific kind of data or metadata. Kosch et al. is restricted to MPEG-7, while in Catteau et al. only a specific vocabulary for learning objects is used. Both Staab et al. and Chen et al. do not prescribe the vocabulary or ontology to be used for data and/or metadata in the system, but assume that a previous ontology creation phase clearly defines which ontology will be used for any metadata in the system. The same is true for Millard et al., who assume that a domain ontology will be modelled in the first phases of the lifecycle, which will then be used to annotate the data in the system (in this case learning objects). Neither Lee and Hong nor McKeever explicitly define what the form of either data or metadata in the systems they describe should be. This means that a clear classification in terms of homogeneous and heterogeneous cannot be made. An exception is Hardman: while it is true that the data in this lifecycle is restricted to some form of media asset, no restrictions are given on the form or terminology used for annotations over this data. Also, even the restriction to media as data is softened somewhat in the sense that annotations are allowed to become media assets in their own right.

Open vs. closed This feature is related to the *homo-/heterogeneous* dichotomy. *Open* and *Closed* describe whether a system allows arbitrary data from the outside to enter its lifecycle, which wasn't initially meant for this particular system. In this sense, the model described in Staab et al. is an example for an open lifecycle, since it includes an explicit import stage for the inclusion of external data. All other models in the survey should be considered closed lifecycles, because they do not provide any import stage. At first sight, this statement does not seem to hold for Hardman, in the sense that the two *planning* steps *premeditate* and *message construction* take their inputs from outside the system. However, this kind of input actually only represents the intent or context of the author for becoming active in the system. To consider this as an example for an open lifecycle would render the feature meaningless, since all lifecycles would then have to be characterised as open.

Centralised vs. distributed This last dichotomy describes the physical nature of a system and the lifecycle of its data and metadata: if it resides in a single, centrally controlled infrastructure, we will call it *centralised*. If it is spread over a a network with no single point of control, we will call it *distributed*. Most systems described in Sect. 2 are examples for centralised lifecycles. One might argue that, if users can access the system online, that those systems are still distributed. However, since the circular flow of data resides in a centralised system, we will still consider them to be centralised. Again, Hardman is an exception: her lifecycle does not describe a concrete system at any one place, but rather the lifecycle of media data as it is processed by various tools. Where those tools reside, how they are connected and how the data flows between them is left open, therefore making the lifecycle a distributed one. Due to their vague nature, both Lee and Hong and McKeever could describe either a centralised or a distributed systems.

Lifecycle type [8] and [10] suggest three general types of lifecycle models, based on how and under what circumstances a new iteration of the lifecycle is reached and how data in the system is changed: (i) *sequential*, (ii) *incremental* and (iii) *evolving*. A *sequential* model [40] (also referred to as *waterfall model*), denotes a kind of lifecycle in which each phase or step can only be reached if the preceding one is completely finished. This also means that a new iteration of the cycle can only be started when all steps have been gone through. In contrast, the *incremental* model allows the start of a new iteration whenever the totality of data in the system is lifted to a new version. This may happen even before the cycle has been finished completely. Finally, the *evolving* model implies that data in the system can change at any time, meaning that new iterations can be started at any time. Consequentially, it would also mean that several iterations of the lifecycle can be ongoing at the same time.

Granularity Since every lifecycle operates on some kind of system, we can also consider how much of that system is affected by a single iteration of the cycle. By going through the individual steps of the cycle, are we manipulating all data in the system or only parts of it? The answer to this question defines the *granularity* of the lifecycle: a model where all data is affected in each iteration has a *coarse* granularity, whereas a

model where only portions of the data are affected has a *fine* granularity. Most lifecycles in the survey should be considered fine-grained models, in the sense that they allow a single iteration for each data object in the system, be it a media asset, a learning object, a document, etc. However, those models which contain a dedicated ontology development phase in the beginning appear to be rather coarse-grained: we first define the domain ontology, then apply it to a corpus of data, then get feedback on the ontology and its application, then revise the ontology based on the feedback, etc. In the survey, examples of this kind of coarse-grained lifecycle are Millard et al. and Chen et al.

There also seems to be a connection between the granularity of a lifecycle and its type. A lifecycle of the evolving type would also tend to be fine-grained (lest we would have the whole system in a constantly changing state, with each state existing alongside the others), whereas a sequential lifecycle would tend to be more coarse-grained.

3.3. Lifecycle roles

Based on the survey in Sect. 2 we propose five different roles (see [37]) in this section, which can be played by actors in the ADLM. Together with the lifecycle phases and features, these roles will aid in describing and classifying various lifecycle models, and ultimately outline the lifecycle model for the Semantic Web. With respect to the relationship between actor and role, it should be noted that the same actor in any particular system for which the lifecycle is applied can play several roles at once. E.g., the same actor could, with their data creator hat on, first plan and create a data object, and then, with their administrator hat on, archive and publish the data.

In addition to the models in the survey, we also consider the roles established in a prescriptive proposal for the lifecycle of URIs on the Semantic Web [4], which are *URI owner* (authority to mint URIs), *statement author* (using URIs in statements) and *consumer* (reading and interpreting an RDF statement).

Each role will typically be involved in specific phases from the lifecycle meta-model, as shown in the overview in Fig. 8.

Ontology designers An actor who is involved in the *ontology development* phase of the lifecycle plays the role of an *ontology designer*. As argued previously, the ontology development phase can often be seen as a preceding lifecycle of its own. For the same reasons, when the ontology is considered data, the ontology designer role could be argued to comprise all other roles within it.

Data creators The role of *data creator* implies that the actor who performs it is creating the primary kind of data in the system at hand. Depending on the system, this could mean the creation of media assets, learning objects, documents, etc. In the literature, this kind of role has been given different names: Greenberg, in her discussion of processes, people and tools, calls this role *content creator*, whereas in Kosch et al., data creators are identified as *content providers/producers* in the tripartite user space. In Booth's proposal, the most suitable mappings for the data creator role are *URI owner* and *statement author*. Within the lifecycle meta-model, data creators partake in the *planning* and *creation* phases.

Metadata creators *Metadata creators* are those actors which annotate the primary data in a system with metadata. This role is called *processing user* in the user space in Kosch et al. and *statement creator* in the proposal by Booth. Greenberg uses the same term as it was chosen for the ADLM, but further distinguishes metadata creators according to their level of professionality. In the lifecycle model, we have chosen to express professionality as a role feature instead. Metadata creators also perform *planning*, after which they either *create* new metadata, or *refine* existing data or metadata.

Obviously, the distinction between data creators and metadata creators will become meaningless when a particular lifecycle doesn't distinguish between data and metadata. In that case, the two roles should be considered identical.

Administrators *Administrators*, in contrast to *creators*, handle data and metadata in the system without actually changing its shape or explicit meaning. In that sense, administrators are responsible for the *archiving*, *publication* and *termination* of data. Other activities, which are not covered in the meta-model, are those outlined in the *control and administration* support process as outlined in McKeever (security, monitoring, etc.).

End users As the last role, *end users* are those actors who do not actively handle the data in the system, but instead passively receive and consume it. According to Kosch et al., end users are involved in "browsing, searching, and consuming" metadata and content. The equivalent in Booth is *consumer*. Within the meta-model, this means that this role is played in the *ac-*

Table 2
Comparison of Metadata Lifecycles — Features

Feature	**Kosch** (Multimedia)	**Hardman** (Multimedia)	**Catteau** (eLearning)	**Millard** (eLearning)	**Chen** (Dig. Library)	**Staab** (KM)	**Lee** (KM)	**McKeever** (WCM)
Distinction Data vs. Metadata	yes	no	yes	yes	yes	no	no	yes
Prescriptive vs. Descriptive	prescriptive	descriptive	prescriptive	prescriptive	prescriptive	prescriptive	descriptive	descriptive
Homogeneous vs. Heterogeneous	homogeneous	heterogeneous	homogeneous	homogeneous	homogeneous	homogeneous	—	—
Closed vs. Open	closed	closed	closed	closed	closed	open	closed	closed
Centralised vs. Distributed	centralised	distributed	centralised	centralised	centralised	centralised	—	—
Lifecycle Type	evolving	evolving	evolving	sequential	sequential	evolving	evolving	evolving
Granularity	fine	fine	fine	coarse	coarse	fine	fine	fine

cess and *external use* phases, but also in the more active *feedback* phase, which allows the closing of the cycle. This gives the end user actor a pivotal role in the lifecycle, which is also played out in the fact that they can change their role and become a data or metadata creator (by re-using the data). By doing this, they will effectively move the data they are consuming back into the *refinement* or *archiving* phase, thus leading to a new iteration of the lifecycle.

3.4. Actor features

Actor professionality Apart from the roles defined in the paragraphs above, each actor playing a role can also be categorised according to their level of professionality. Borrowing the suggestion made in Greenberg, we propose two levels of professionality, which can be seen as the extreme points on a vector of professionality: *professional* and *community or subject enthusiasts*. Greenberg only applies these attributes to metadata creators, whereas in this model they will be applied to actors performing any of the five roles. *Professionality* is a function of individual actors paired with a role. I.e., an actor might be a *community or subject enthusiast* as a *Data Creator*, but *professional* as an *Administrator*.

Actor humanness Any of the roles defined for data lifecycles can be performed by actors which are either *human* or *machine* agents. This categorisation is also highlighted by Greenberg, but restricted to metadata generation. In the model proposed here, humanness can be applied to actors performing any role of the lifecycle. Situations which involve any kind of semi-automatic operation should be modelled with two actors — one human and machine[9].

3.5. Metadata features

As a final area of classification, we will present some features that have been suggested for the description of metadata in the literature.

Authoritative vs. non-authoritative Where data and metadata are distinguished, metadata can be either authoritative or non-authoritative. Authoritative metadata will be used in preference over all other, non-authoritative metadata. The exact definition of both terms depends on the usage context; however, generally speaking metadata is considered authoritative if it comes directly from the author or source of a piece of data. E.g., authoritative metadata about a digital image would be the metadata that comes directly from the creator of the image. On the other hand, metadata is considered non-authoritative if it originates from any source other than the original author of the data. E.g., tags created through the means of social tagging (i.e., by the community) would be considered non-authoritative metadata. For a discussion of authoritative metadata as part of Web architecture see [11]. As an example for the relevance of this feature, [20] propose the use of T-Box reasoning over authoritative statements only to tackle scalability issue in Web-scale reasoning.

Content-independent vs. content-dependent This dichotomy is introduced as part of a classification scheme for metadata in [22] and classifies metadata according to its content dependency. Examples for *content-independent* metadata are modification date or author of a document, a logo, or sensor information for an image. *Content-dependent* metadata on the other hand is directly related or even derived from the content of the data it is related to. Examples for this kind of metadata are document size or image resolution.

Direct Content-based vs. content-descriptive A further classification of *content-dependent* metadata from Kashyap and Sheth describes how closely metadata is based on its data. *Direct content-based* metadata is directly derived from its data, such as a document full-text index or document vectors. *Content-descriptive* metadata on the other hand is an indirect description of data, such as textual descriptions or markup.

Domain-independent vs. domain-specific Drilling down further in their classification of metadata, Kashyap and Sheth suggest this dichotomy to describe the domain dependency of *content-descriptive* metadata. Metadata which can be used to describe other data regardless of its domain is called *domain-independent*. A typical example would be format-related markup such as HTML. In contrast, *domain-specific* metadata is applicable only depending on the subject domain of its data. A domain ontology or domain-specific markup are examples.

A similar distinction is discussed in [36]. The authors there divide the domain of metadata into *structural metadata* and *content metadata*, where the former describes the internal and external structure of its data (and is therefore content-independent), whereas

[9] Unless, of course, the actor is a cyborg. In this case, a *human-machine* category should be added to the model.

the latter describes the content of its data (and is therefore content-specific). Also Kosch et al. make this distinction by using the two opposing terms "metadata for content adaption" (structural) and "metadata for search and retrieval" (content). Occasionally, domain-independent metadata is further differentiated into *structural* and *administrative* metadata (e.g., [31]).

4. Applying the ADLM to the Semantic Web

In the following, we will revisit the different phases and features proposed in previous sections and discuss whether and how they are realised on the Semantic Web.

4.1. Semantic Web lifecycle phases

Ontology Development Ontologies are a central component of a self-describing, semantic Web. Based on the evidence found in the survey, the definition of the ontology development phase in the previous section stated that this phase should either be considered to be outside the lifecycle model (since it concerns a meta layer and therefore doesn't directly consider the flow of data itself), or to be its own lifecycle altogether. While this is still true for the Semantic Web, there is, however, a slight modification: Vocabularies and ontologies on the SW on the one hand and primary data (RDF statements using the terms defined in the ontologies) on the other — or TBox and ABox in description logic terms — cannot be distinguished with respect to their form. Any document or set of statements can contain any mixture of definitions of ontological terms and instances of them. Therefore, data in the ontology development phase has to be interpreted in two ways: (i) Ontological data in the sense of a conceptualisation of a domain. In this sense, ontology development is a separate lifecycle, with its own set of phases and features, e.g., those defined in [8]. Indeed, we shall consider it to be a separate instance of the ADLM, proceeding that of the primary (or instance) data. (ii) Ontological data in the sense of their nature as RDF statements. In this sense, ontological data is indistinguishable from other data on the Semantic Web and part of the same lifecycle. Which of the two interpretations applies, depends on the specific requirements of the scenario for which the ADLM is used.

Planning With respect to planning, the lifecycle of data on the Semantic Web is no different from that of any other system. Planning must precede any creation or refinement of data. It does not touch the data itself, and is therefore outside of the system. Examples of planning include (i) the decision process that leads to the actual creation of data, (ii) defining a URI pattern for any new resources that shall be created, or (iii) setting up the necessary technical infrastructure that might be needed for it, such as servers or tools to transform source data from other formats into RDF.

Creation In terms of the ADLM, data creation means introducing data into the lifecycle's system which did not exist *within the system* before. For the purpose of this paper, we will say that any data which is not a set of RDF statements, does not qualify as data within the Semantic Web. In a looser interpretation of "Semantic Web", one could of course argue that data in various other formats can be integrated into the Semantic Web, be it by extraction, transformation or reference. However, for the purpose of this discussion, such data will only be considered once it has been expressed in terms of the system, i.e., as RDF statements. Creation of data on the SW in our model therefore means the creation of new RDF statements.

Refinement Refinement means to change or make additions to existing data in some way. An example would be updating an RDF description of someone's bibliography with references to new papers or articles. Also the process of ontology mapping should be classified as refinement, in the sense that one makes additional statements about ontology terms. Since, within the model discussed in this chapter, all data is RDF, and since no difference is made between data and metadata (see Sect. 4.2), refinement and creation on the Semantic Web are technically the same. *Making additions* to existing data could tentatively mean making additions to an RDF statement. This is not possible, however, since RDF statements are immutable[10]. Making additions therefore has to mean creating new statements about a resource that is part of a previously existing statement. If the focus is on an RDF graph (a set of statements), rather than an individual statement, making additions then means creating a new statement to add to the graph. This point of view also captures the notion of database updates, applied to RDF. In

[10]Changing an RDF statement would create a new one, which could not be explicitly related to the original one. There is no concept of versioning in RDF.

other words, updates on the SW are the addition of statements to existing RDF graphs. In both cases — statement-level and graph-level —, refinement is therefore a special case of creation.

However, for some contexts it is still useful to keep a distinction between the two phases. When a user creates a description of a real world entity in RDF, this description will in most cases be a set of statements. Conceptually, the user might consider creating this set of statements as *one* act of creation, saying e.g. "I created an RDF description of Galway". When, at a later stage, this RDF graph is extended by adding more statements about Galway, the user might consider this to be an act of refinement, clearly distinct from the initial creation, saying "I have refined the description of Galway". In other words: at statement level, there is only creation, while at graph level, there are creation and refinement.

Inferencing and *reasoning*, as central concepts of the Semantic Web, would be considered examples of creation and/or refinement: the application of inference rules implies creating new statements based on other, pre-existing statements. Since the new statements will most likely be related to the resources involved in the existing ones, inference will more often than not be an example of refinement.

Lifecycle aspects such as *provenance* and *trust* are essentially the addition of metadata to the lifecycle's primary data, and should as such also be interpreted as a type of refinement. Alternatively, such metadata could have its own lifecycle, which is linked to the primary data lifecycle. This is similar to the way ontology development can be its own lifecycle.

Archiving The archiving phase in the SW data lifecycle constitutes *making data persistent*. In concrete terms, this can e.g. mean to serialise data in a particular RDF serialisation format, or storing and indexing it in one of the many RDF databases available. In more general terms, archiving means to physically prepare the data in a way that allows it to be *published* (see next phase).

Technically, creating or refining data and archiving are closely related and hard to distinguish in the model. On the one hand, it could be argued that data, i.e., RDF statements, are created in a transient, non-persistent way, e.g., by simply thinking them or as an in-memory representation. However, there will hardly be any application that will facilitate creation but not archiving of some form, or an actor on the lifecycle that is responsible for the former, but not the latter. It is perfectly feasible to say that a user creates RDF data in a file, which is then loaded into a database by another user. However, this only means that data has been created and archived, and then archived a second time. The conclusion is that creation/refinement cannot occur on its own in any realistic scenario, whereas archiving can. It is still beneficial to distinguish between the two stages, e.g., when the model is applied for the analysis of a single application, in order to define a separation of concerns within it (which part of the code or UI is concerned with creation, and which is concerned with archiving).

Furthermore, there are cases where data is created and published immediately, without ever being archived. Examples for this are data wrappers which generate RDF on the fly from a source format (e.g., a GRDDL transformation), on inference engines which compute additional statements of an RDF graph based on RDFS or OWL entailment rules on the fly, i.e., without ever materialising those statements physically.

Publication Publication of data on the Semantic Web means to make it available on the World-wide Web. In other words, it means making it available at an HTTP URI on a Web server. In the case of data which has previously been archived, this can either happen in the form of a file that is served, or an interface to an RDF database (such as a SPARQL endpoint). Alternatively, as described in the previous section, RDF data can be created and published in one seamless step.

Access From a technical perspective, the access phase in our model on the Semantic Web can be defined as the dereferencing of an HTTP URI by an agent, given that SW data has been published to and is served at that URI. Dereferencing here means to send an HTTP request to a server and receive a HTTP response in return. Access is the flip side of publication.

Access can occur in two different ways, which determine how the lifecycle is continued. If an agent accesses a URI on the Semantic Web and retrieves RDF data, this means it stays within the system. In this case, the data can then be refined (changed), archived and published again, possibly at a different location than before. Alternatively, an agent could also access a SW URI and receive an answer in a different format, such as an HTML page. In this case, the data would be exposed outside the system and enable external use.

External Use External use means the utilisation of data outside the system of the lifecycle. For the Semantic Web, this means to use RDF outside it (use *within* the Semantic Web means the flow of data through

many of the other phases of the lifecycle: data is used on the SW when it is refined, when it is archived, published, or accessed). Examples of external use on the Semantic Web would be the aggregation of data from various sites in order to answer a particular query for an external agent, or the conversion and import of data into an external application, such as a desktop address book application. The latter scenario is described in [35].

Strictly speaking, even the rendering of a Web page in a browser derived from RDF data already constitutes a case of external use. However, in the case of an HTML page with embedded RDFa, an interesting situation arises, in which the same document is a hybrid of published Semantic Web data and external use as non-Semantic Web data. Also, it should be noted that external use can take place without the user actually realising it. E.g., a user could direct their Web browser at the URI `http://data.semanticweb.org/conference/iswc/2008`. Through content-negotiation, this particular server would determine that the browser requires HTML and redirect to `http://data.semanticweb.org/conference/iswc/2008/html`, which is an HTML document. For the user, it would not be obvious that they have just accessed and used SW data.

Feedback In the feedback phase end users give comments on the data they have accessed and used. In the ADLM, it is suggested that feedback implies a systems which has a central authority to which the feedback can be addressed. Following this notion, there would be no feedback on the Semantic Web as such, since there is no central authority. However, it is obvious that the Semantic Web as a system at large has many different sub-systems which are the providers of individual services or datasets. Therefore, while users of the Semantic Web could not give feedback to the overall system (just as Web users cannot give feedback to the Web as such), they could certainly give feedback to the central authority of each sub-system. This feedback can then trigger actions on the side of the service provider, such as the termination or refinement of data.

Termination Termination, finally, means to remove data from the system. In a distributed system like the Semantic Web (just like on the World-wide Web), where data is constantly crawled, indexed and duplicated by various players, it is not possible (or at least very difficult) to completely terminate any piece of data, i.e., remove it from the system without any trace. E.g., while the publishers of a particular dataset on the SW might decide to remove a number of RDF statements from their site, it is very likely that this data has previously been crawled by a different service, and will therefore still exist in some form after the termination.

Table 3
Features of the Semantic Web data lifecycle

Distinction Data vs. Metadata	*no*
Prescriptive vs. Descriptive	*descriptive*
Homogeneous vs. Heterogeneous	*heterogeneous*
Closed vs. Open	*open*
Centralised vs. Distributed	*distributed*
Lifecycle Type	*evolving*
Granularity	*fine*

4.2. Semantic Web data lifecycle features

Wrapping up the discussion on applying the ADLM to the SW, this section will characterise the lifecycle of data on the Semantic Web in terms of the features defined in Sect. 3.2. As an overview, all features are summarised in Tab. 3.

Distinction data vs. metadata We have stated that, for the purpose of this paper, we only consider RDF statements as data. Furthermore, in RDF semantics [18] there is no conceptual difference between data and metadata. Consequently, the lifecycle model is characterised as having *no distinction between data and metadata*.

Prescriptive vs. descriptive The lifecycle model is *descriptive*, simply observing the properties of the Semantic Web as they are perceived by the authors of this paper, and classifying them in the terms of the ADLM. It is not prescriptive because it does not propose any rules of how the lifecycle of data on the Semantic Web should look like.

Homogeneous vs. heterogeneous The data in the lifecycle's system is *heterogeneous*, because no assumption is made about the schema or ontology which defines the various resource types, such as classes, instances or properties. In fact, agents do not have to know any data's schema. Data can even be completely schema-less and still be useful. One could assume the opposite standpoint and argue that the data is in fact homogeneous, because only RDF is considered. However, this only means that the data is homogeneous syntactically. It is still heterogeneous semantically, which is what is considered by this feature.

Closed vs. open With the restriction that any data entering the system first has to be expressed as RDF, the Semantic Web is an *open* system. Any data can be integrated, as anything can be said about anything [24], which means that there are no restrictions on the scope of domains or topics which can be discussed.

Centralised vs. distributed The underlying architecture of the the Semantic Web is that of the World-wide Web, which has no place for any one central authority. Consequently, and since it does not add any such authority itself, the Semantic Web is a *distributed* system.

Lifecycle type The lifecycle type of the model proposed here is *evolving*, since there is no requirement for data to pass through the lifecycle completely before a new iteration can be started. In fact, there is no such thing as "complete" iteration in the model. RDF data can be created, published, accessed, refined, archived, published, etc. These phases can be passed in many different orders and, in principle, for an infinite number of times.

Granularity Finally, the Semantic Web data lifecycle is *fine-grained*, meaning that the cardinality of the set of statements considered by any particular instance of the lifecycle can be as small as 1. The lifecycle is equally applicable to individual statements and large graphs, which can both pass through any of the phases proposed in the ADLM.

4.3. Using the ADLM

Concluding the application of the ADLM to the Semantic Web, two examples shall illustrate how the model can be used in practice, by grounding discussion in a precise terminology.

Comparing applications A brief look at two cases of SW-enabled websites will illustrate how the ADLM can be applied to analyse differences in the approach to the publication of data on the Semantic Web. Version 7 of the Drupal content management system features built-in RDFa functionality, and so a means to publish data on the Semantic Web. The "Semantic Web Dog Food" (SWDF) site for conference data and metadata (details in [34]) also publishes RDF to the Semantic Web, and is based on Drupal. At first sight, the two implementations seem to be doing more or less the same thing. However, the SWDF approach to publication and the Drupal 7 RDFa functionality are fundamentally different: where SWDF uses Drupal to publish pre-existing RDF to both the Semantic Web (in its original RDF form) and the traditional eye-ball Web (as HTML), Drupal 7 transforms and publishes the non-RDF contents of its own database to the Semantic Web as RDFa. Using the ADLM, we can more precisely describe SWDF as covering the *creation* phase (conference data is created in file format), the *archiving* phase (data is archived in an RDF store) and the *publishing* phase (data is published through Drupal). These phases are visited in sequence, as shown in Fig. 9. In Drupal 7, on the other hand, data is created *on the fly* from the Drupal database, and then injected as RDFa in a Drupal page. Again employing the ADLM's terminology, we can explain that *on the fly* more precisely means that the *publication* phase follows the *creation* phase directly, with no intermediate *archiving* in play, as also shown in Fig. 9.

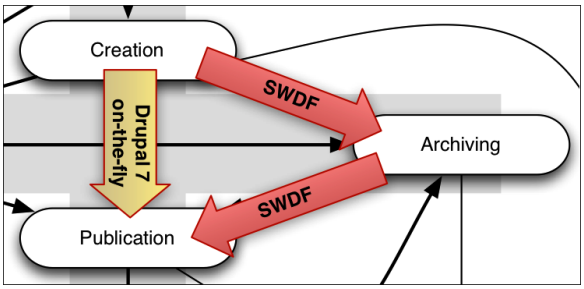

Fig. 9. Different approaches to *publication* on the SW

In a less detailed way, also [17] (see Sect. 2.4) provides an example of how a lifecycle model (here a linked data lifecycle) can be used to classify and group applications, with the purpose of giving an overview of a broader field or the work done within a research group.

Explaining paradigm shifts [16] observes a (possible) paradigm shift from the (i) traditional publishing and usage of data to (ii) publishing and usage of data according to the principles of linked data. In both cases, data publishers use transformation templates to publish data from their internal databases to a webpage. However, in (i) the transformation is to traditional HTML (meaning that software agents have to employ screen scraping to re-discover structured data on the webpage), while in (ii) the transformation is to HTML+RDFa (meaning that software agents can take structured data directly from the page).

This paradigm-shift can be nicely described using the ADLM: In the traditional scenario (i) the SW's data lifecycle is only entered through the software agent's

screen scraper, which performs the *creation* phase of the lifecycle. On the other hand, in the linked data approach (ii) the lifecycle is already entered on the data publisher's side, where the transformation template *creates* SW data, which is then *published* on the fly (as in the previous example). From there, a software agent is ready to simply *access* the data. In both cases, the software agent can then proceed to the *archiving* (e.g., a crawler), *refinement* (e.g., an aggregated query) or *external use* (e.g., visualisation in an infographic) phase.

5. Conclusion

The aim of this paper was two-fold: (i) to provide a wide overview of different approaches to lifecycle modelling of data-centric domains, and (ii) to establish a terminological framework and conceptual model for data and metadata lifecycles in data-centric domains. The overview was presented in the form of a survey of a significant number of different lifecycle models from the literature, ranging across different domains. The survey itself functions as a first port of call for researchers and developers aiming to design a lifecycle for a particular domain, by providing an overview and inspiration over typical modelling approaches and practices.

Based on the survey, as well as additional literature, five areas of classification for data lifecycles have been identified and discussed, which together form the terminological framework and model, called the Abstract Data Lifecycle Model (ADLM): lifecycle phases, lifecycle features, lifecycle roles, actor features and metadata features. None of the individual models covered in the survey suffices as a generic model over any data-centric domain; the ADLM can fulfil this task. In addition to its primary aspects (phases, features and roles), the ADLM also enables to model secondary aspects of data lifecycles, such as versioning (through repetition of the cycle) or provenance and trust (as metadata creation through the refinement phase).

The function of the ADLM as a meta model is to provide a means to classify, compare and relate other lifecycle models, as well as to provide the basis to construct new lifecycle models for other data-centric domains, applications or use cases. In the third part of the paper, this kind of use is illustrated by applying the model to the Semantic Web. This SW instance of the ADLM serves both to support the general purpose of defining a conceptual framework for the SW, as well as to show how application developers or researchers can use the framework to illustrate design issues or ground discussion in a common terminology.

With respect to other proposed lifecycles for the Semantic Web or Web of Data, no comprehensive proposal is known to the authors. However, a number of SW-related lifecycles have been discussed as part of Sect. 2.4 on lifecycles for knowledge and content management, as well as in Sect. 4.3 on using the ADLM.

Acknowledgments

The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Líon-2) and (in part) by the European project NEPOMUK No. FP6-027705 and (in part) by the European project FAST No. FP7-216048.

References

[1] *OED Online*. Oxford University Press, June 2011. http://www.oed.com/viewdictionaryentry/Entry/108098.

[2] Tim Berners-Lee. Linked data, 2006. URL http://www.w3.org/DesignIssues/LinkedData.html.

[3] Tim Berners-Lee. Artificial intelligence and the Semantic Web. Talk, 2006. http://www.w3.org/2006/Talks/0718-aaai-tbl/ 27/05/2009.

[4] David Booth. The URI lifecycle in Semantic Web architecture, July 2009. http://dbooth.org/2009/lifecycle/ 29/01/2012.

[5] O. Catteau, P. Vidal, and J. Broisin. A generic representation allowing for expression of learning object and metadata lifecycle. In Kinshuk, Rob Koper, Piet Kommers, Paul Kirschner, Demetrios G. Sampson, and Wim Didderen, editors, *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade, The Netherlands*, pages 30–32, Los Alamitos, CA, USA, July 2006. IEEE Computer Society.

[6] Ya-Ning Chen, Shu-Jiun Chen, and Simn C. Lin. A metadata lifecycle model for digital libraries: methodology and application for an evidence-based approach to library research. In *World Library and Information Congress: 69th IFLA General Conference and Council*, August 2003. http://archive.ifla.org/IV/ifla69/papers/141e-Chen_Chen_Lin.pdf 29/01/2012.

[7] David Crystal. *The Cambridge Encyclopedia of Language*. Cambridge University Press, second edition, 1997.

[8] Mariano Fernández, Asunción Gómez-Pérez, and Natalia Juristo. METHONTOLOGY: From ontological art towards ontological engineering. In *Workshop on Ontological Engineering at AAAI97, Stanford, USA*, Spring Symposium Series, 1997.

[9] Mariano Fernández, Asunción Gómez-Pérez, and Mariá Dolores Rojas Amaya. Ontology's crossed life cycles. In *Lecture Notes in Artificial Intelligence*, number 1937, pages 65–79. Springer-Verlag Heidelberg, 2000.

[10] Mariano Fernández-López and Asunción Gómez-Pérez. Overview and analysis of methodologies for building ontologies. *Knowedge Engineering Revue*, 17(2):129–156, 2002.

[11] Roy T. Fielding and Ian Jacobs. Authoritative metadata. Tag finding, W3C, April 2006. URL `http://www.w3.org/2001/tag/doc/mime-respect-20040225`.

[12] Aurona Gerber, Alta van der Merwe, and Andries Barnard. A functional Semantic Web architecture. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, volume 5021 of *LNCS*, pages 273–287. Springer, June 2008.

[13] Asunción Gómez-Pérez, Mariano Fernández, and Antonio J. de Vicente. Towards a method to conceptualize domain ontologies. In *Workshop on Ontological Engineering at ECAI'96, Budapest, August 13, 1996*, pages 41–51, 1996.

[14] Jane Greenberg. Metadata generation: Processes, people and tools. *Bulletin of the American Society for Information Science and Technology*, 29(2), 2003.

[15] Lynda Hardman. Canonical processes of media production. In *MHC '05: Proceedings of the ACM workshop on Multimedia for human communication*, pages 1–6, New York, NY, USA, 2005. ACM.

[16] Michael Hausenblas. Data lifecycle, September 2009. Blog Post: `http://webofdata.wordpress.com/2009/09/14/data-lifecycle/`.

[17] Michael Hausenblas. Linked data research centre — projects, research, applications. Internal presentation at DERI, May 2011.

[18] Pat Hayes. RDF semantics. Recommendation, W3C, February 2004. `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/` 29/05/2009.

[19] Sarah Higgins. The DCC curation lifecycle model. *The International Journal of Digital Curation*, 3(1):134–140, June 2008.

[20] Aidan Hogan, Andreas Harth, and Axel Polleres. SAOR: Authoritative reasoning for the Web. In John Domingue and Chutiporn Anutariya, editors, *3rd Asian Semantic Web Conference, ASWC 2008, Bangkok, Thailand*, volume 5367 of *LNCS*, pages 76–90. Springer, December 2008.

[21] Ian Jacobs and Norman Walsh. Architecture of the World Wide Web, Volume One. W3C Recommendation. Recommendation, W3C, December 2004. URL `http://www.w3.org/TR/2004/REC-webarch-20041215/`. `http://www.w3.org/TR/2004/REC-webarch-20041215/`.

[22] Vipul Kashyap and Amit Sheth. Semantic heterogeneity in global information systems: The role of metadata, context and ontologies. In M. Papazoglou and G. Schlageter, editors, *Cooperative Information Systems: Current Trends and Directions*, pages 139–178. Academic Press, June 1996.

[23] Haim Kilov. From semantic to object-oriented data modeling. In Peter A. Ng, C. V. Ramamoorthy, Laurence C. Seifert, and Raymond T. Yeh, editors, *Proceedings of the First International Conference on Systems Integration, Morristown, NJ, USA, April 1990 (ICSI1990)*, pages 385–393. IEEE Computer Society, April 1990.

[24] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. Recommendation, W3C, February 2004. `http://www.w3.org/TR/rdf-concepts/` 28/05/2009.

[25] Harald Kosch, László Böszörményi, Mario Döller, Mulugeta Libsie, Peter Schojer, and Andrea Kofler. The life cycle of multimedia metadata. *IEEE MultiMedia*, 12(1):80–86, 2005.

[26] David B. Leake and David C. Wilson. Categorizing case-base maintenance: Dimensions and directions. In Barry Smyth and Padraig Cunningham, editors, *Advances in Case-Based Reasoning, 4th European Workshop, EWCBR-98, Dublin, Ireland, September 1998, Proceedings*, volume 1488 of *LNCS*, pages 196–207. Springer, 1998.

[27] Sang M. Lee and Soongoo Hong. An enterprise-wide knowledge management system infrastructure. *Industrial Management & Data Systems*, 102(1):17–25, 2002.

[28] Daniel D. McCracken and Michael A. Jackson. Life cycle concept considered harmful. *SIGSOFT Softw. Eng. Notes*, 7(2): 29–32, 1982.

[29] Susan McKeever. Understanding Web content management systems: evolution, lifecycle and market. *Industrial Management & Data Systems*, 103(9):686–692, 2003.

[30] Noah Mendelsohn. The self-describing web. Tag finding, W3C, February 07 2009. `http://www.w3.org/2001/tag/doc/selfDescribingDocuments-2009-02-07.html` 03/03/2009.

[31] METS. <METS> Metadata encoding and transmission standard: Primer and reference manual. Technical report, Library of Congress, April 2010.

[32] David E. Millard, Feng Tao, Karl Doody, Arouna Woukeu, and Hugh C. Davis. The knowledge life cycle for e-learning. *International Journal of Continuing Engineering Education and Lifelong Learning: Special Issue on Application of Semantic Web Technologies in E-learning*, 16(1/2):110–121, April 2006.

[33] Felix Mödritscher. Semantic lifecycles: modelling, application, authoring, mining, and evaluation of meaningful data. *International Journal of Knowledge and Web Intelligence*, 1(1/2): 110–124, 2009.

[34] Knud Möller. The Semantic Web conference metadata project. In *Lifecycle Support for Data on the Semantic Web*, pages 73–111, September 2009. PhD Thesis, NUI Galway, Ireland.

[35] Knud Möller and Stefan Decker. Harvesting Desktop Data for Semantic Blogging. In *1st Workshop on the Semantic Desktop at ISWC2005, Galway, Ireland, Proceedings*, pages 79–91, November 2005.

[36] Knud Möller, Uldis Bojārs, and John G. Breslin. Using Semantics to Enhance the Blogging Experience. In John Domingue York Sure, editor, *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*, volume 4011 of *LNCS*, pages 679–696. Springer, June 2006.

[37] Francis G. Mossé. Modeling roles — a practical series of analysis patterns. *Journal of Object Technology*, 1(4):27–37, September-October 2002.

[38] Vit Nováček, Siegfried Handschuh, Diana Maynard, Loredana Laera, Sebastian Ryszard Kruk, Max Völkel, Tudor Groza, and Valentina Tamma. Report and prototype of dynamics in the ontology lifecycle. Deliverable 2.3.8v1, Knowledge Web, January 2006.

[39] Vit Nováček, Sebastian Ryszard Kruk, Maciej Dabrowski, and Siegfried Handschuh. Extending Community Ontology Using Automatically Generated Suggestions. In David Wilson and Geoff Sutcliffe, editors, *Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Confer-*

ence, May 7-9, 2007, Key West, Florida, USA. AAAI Press, May 2007.

[40] W. W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings, 9th International Conference on Software Engineering (ICSE '87), March 30 - April 2, 1987, Monterey, California, USA*, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[41] David J. Schultz. IEEE standard for developing software life cycle processes. Technical Report 1074-1995, IEEE, 1996. `http://ieeexplore.ieee.org/servlet/opac?punumber=3513` 29/01/2012.

[42] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems, Special Issue on Knowledge Management*, 16(1):26–34, January 2001.