Treating the End of the Data Life Cycle as a First-Class Citizen in Data Engineering

Daniel Tebernum
Fraunhofer ISST, Germany, daniel.tebernum@isst.fraunhofer.de

Falk Howar
TU Dortmund, Germany, falk.howar@tu-dortmund.de

Treating the End of the Data Life Cycle as a First-Class Citizen in Data Engineering

Research Paper

Daniel Tebernum[1] and Falk Howar[2]

[1] Fraunhofer ISST, Data Business, Dortmund, Germany
daniel.tebernum@isst.fraunhofer.de
[2] TU Dortmund University, Chair for Software Engineering, Dortmund, Germany
falk.howar@tu-dortmund.de

Abstract. Evolving regulatory frameworks, digital decarbonization, or new management challenges due to exponential data growth are bringing end-of-life data management to the forefront and making it an important building block in data engineering. However, the end of the data life cycle and in particular its management has received little attention in literature and practice. We argue this is partly due to the lack of an overarching model that creates a common understanding that data engineering experts and practitioners can build upon. We present Destroy Claims, which provides a standardized and comprehensive description of the end of the data life cycle and a corresponding architectural proposal for its integration. We demonstrate the feasibility of the solution through a use case inspired by practice, an evaluation gathered from expert discussions, and a survey. The results indicate that the proposed solution is a promising approach to support end-of-life data management in practice.
Keywords: End of Data Life Cycle, Data Deletion Management, Data Model, Data Deletion Architecture, Data Catalog.

1 Introduction

Both research and practice have established a widely accepted view that efficient management of data can have a positive impact on business agility (Otto 2015, Tallon et al. 2013). However, current considerations in data engineering rarely address the end of the data life cycle, and even more so, they fail to encompass its management (Tebernum et al. 2021, 2023). With only 121 relevant publications in 23 years, data deletion has not received much attention, even across communities (Tebernum & Howar 2023). This is surprising as there are many benefits to addressing and professionalizing end-of-life data management.

First, evolving regulatory frameworks, i.e., laws and standards, highlight the need for data deletion management (see e.g., (European Commission 2016) and NIST 800-88). Second, the reduction of cost is of high relevance with regard to rising storage costs (see e.g., Google Cloud (2023*a*)). Next, the topic of digital decarbonization has gained importance (Jackson & Hodgkinson 2022) with events like the European energy crisis (Council of the EU 2022), and the fact that large parts of the data that are collected and stored only get used once (Trajanov et al. 2018) creates awareness for a more effiecient use of limited resources. In addition, deleting data of poor quality is crucial to prevent unwanted propagation inside a company and possibly resulting negative influence on processes and products (Chae et al. 2014). Furthermore, privacy and security should be considered. For instance, data may be irretrievably and verifiably deleted before old storage media are sold (Klonowski et al. 2019), or must be deleted to prevent social engineering attacks. Consequently, also in the domain of data ecosystems and data spaces, data sovereignty considerations may require data deletion (Jung et al. 2022, p.133). Finally, it is essential to consider technical factors such as the requirement to delete corrupted, invalid, or noisy data (Othon et al. 2019) and the need for efficient computations through the removal of unnecessary data (Gao et al. 2019, Reardon et al. 2012, Lin et al. 2009, Pachpor & Prasad 2018).

Consequently, the question arises why, despite the variety of relevant reasons for end-of-life data management, the subject is underrepresented in literature and practice. With the help of expert interviews, we identified two relevant factors: (1) in general, there is a missing unified understanding of the end of the data life cycle; (2) particularly in practice, there is a lack of comprehensive solutions on which to build for managing the end of the data life cycle. Therefore, the following research questions arise:

- RQ1: *How can the end of the data life cycle be modelled comprehensively?*
- RQ2: *How can such a model and its interpretation be integrated into a software architecture to support end-of-life data management?*

To address these research questions, two artifacts are developed. Our first artifact is a model that can describe the end of the data life cycle in a uniform way. Using this model, a common understanding, planning capabilities, and an automated execution of the end of the data life cycle can be achieved. To the best of our knowledge, there is no purpose-built solution that comprehensively and holistically models the end of the data life cycle. Our second artifact is an architectural proposal for the integration of the model into software. With our artifacts we want to support the establishment of the end of the data life cycle as first-class citizen in data engineering.

The remainder of this paper is organized as follows. First, in Section 2, we discuss the research methodology and describe our research process. In Section 3 we describe our two artifacts in detail. Section 4 demonstrates the applicability in the context of a real-world use case. A comprehensive analysis of the evaluation of our artifacts, including discussions on limitations and future work are part of Section 5. Section 6 addresses related work. Finally, we summarize our results and describe methodological limitations.

2 Methodology

The results presented in this paper were developed utilizing the Design Science Research (DSR) Methodology. DSR is widely used and shaped by the Information Systems (IS) community (Österle et al. 2010). It focuses on the derivation and iterative development of artifacts. This work is based on the DSRM Process Model by Peffers et al. (2007) (see Figure 1).

Motivation: see Section 1.

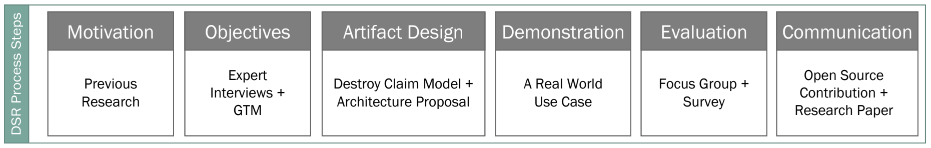


Figure 1. Research process based on Peffers et al. (2007, p.54)

Objectives: To identify what can be leveraged to support end-of-life data management, we sought a real-world perspective. We conducted interviews with experts having between 5 and 15 years of experience. Among the interviewees were four software developers from the industry focussing on data engineering in mobile communication, automotive, machine learning, banking, and pharmacy. We also interviewed two business engineers from the research community with a focus on cloud platforms and data management to reflect a perspective that is not purely technical. Semi-structured interviews, which lasted one hour on average, were carried out through video calls and were subsequently recorded and transcribed. The main guiding questions were "*What comes to your mind about data deletion?*", "*When do you delete data?*", and "*What would push you to delete data?*". In addition, we prepared a miroboard where we presented previous research results and also our own ideas for discussion. We evaluated the interviews by applying an open coding on the transcripts according to the Grounded Theory Methodology (Glaser et al. 1968, Corbin & Strauss 1990). We then isolated the essence of the open codes in axial codes. The evaluation of the interviews identified two objectives to be addressed. First, it must be possible to describe the end of the data life cycle in a way that supports an unified understanding. This objective is supported by increased occurrence of the axial codes *standard*, *model* and *transparency*. Second, to support end-of-life data management, a system architecture must be designed that addresses data deletion as a first-class citizen. We derived this objective because of an increased occurrence of the codes *architecture*, *component*, and *automatic*. These two objectives are reflected in RQ1 and RQ2. The list of codes are available at Zenodo.[1]

Artifacts Design: We developed two types of artifacts. In addition to our own ideas, insights from the expert interviews and our previous publication (Tebernum & Howar 2023) were also used for the design of the artifacts. The first artifact (A1) is the Destroy Claim Model, which we specifically designed to model and standardize the end of the data life cycle. This artifact of type model pays directly into RQ1. The second artifact (A2) is an architectural proposal on how to manage, distribute, and automate the end of the data life cycle in a real environment. This artifact of type architecture pays directly into RQ2.

Demonstration: The artifacts are utilized in a real-world use case. Through this, we show that the artifacts can be used for certain classes of problems. The demonstration was carried out using prototypical implementations. The use case demonstrated with screenshots is available online.[1] Further details are described in Section 4.

Evaluation: As the basis for our evaluation, we chose (1) focus groups (Hollander 2004) to foster a rich discussion and (2) the Strategic DSR Evaluation Framework

[1] `https://doi.org/10.5281/zenodo.8046369`

(Venable et al. 2012) as a structured feedback mechanism. First, we presented our artifacts to nine data and software engineering experts. Among these were the four software engineers interviewed in the objectives DSR phase. We were able to recruit five additional software engineers ranging from 3 to 20 years of experience. Three from the industry with a focus on software architectures and data pipelines and two from research. A whiteboard was prepared for the virtual discussion rounds, which were performed in groups of three. On the whiteboard, previous publications and the existing research results of this work were shown. The experts were given detailed explanations of both the technical details and the demonstrated use case. Then a discussion was initiated in which the experts were invited to address ideas and problems. These were collected and refined on the whiteboard. Second, for structured feedback, the experts were given a questionnaire with 37 5-point Likert scale questions derived from the taxonomy of DSR evaluation methods by Prat et al. (2015). The results of the discussion and questionnaire are discussed in Section 5.

Communication: Communication is done through this paper, which informs the scientific community about our research project. Also, results were published as an open-source contribution on GitHub (Tebernum 2023, Tebernum & Atamantschuk 2023).

3 Contribution

3.1 The Destroy Claim Model (A1)

This artifact contributes directly to RQ1. The aim is to comprehensively describe the end of the data life cycle for arbitrary data. In one of our previous research papers, we developed a data deletion taxonomy, which is derived from an extensive systematic literature review (Tebernum & Howar 2023). We used this as a basis to iteratively develop the model. In short, the questions of *what*, *when*, *where*, *how*, *why*, and *who* must be answered. Based on this, we developed an abstract model. It can be derived into concrete persistence formats according to one's own needs. E.g., a specific derivation in JSON has been generated and published on GitHub (Tebernum 2023). Figure 2 shows the final iteration of the artifact.

Due to space constraints, the model is only described in general terms. More details can also be found on GitHub. The *DestroyClaim* class can be seen as the entry point of the model. It contains higher-level properties that pursue several purposes. On the one hand, there are descriptive properties that should enable human actors to better understand the Destroy Claim (see Table 1). On the other hand, it contains instructions on how to interpret the Destroy Claim (see Table 2).

Another important aspect is the specification of a deletion reason. We were able to identify a set of 52 reasons *why* one would want to delete data (see Tebernum & Howar (2023)) and standardized them.[2] It is possible to have none or several *destroyReasons* and to also define custom ones.

Extension is a concept that greatly increases the flexibility of the model. It quickly became apparent that the use cases for deleting data are extremely heterogeneous and

[2] `https://github.com/DaTebe/destroyclaims/blob/9a8a6e5e432312175cc439f333c45620924400fc/docs/destroy-reasons.md`

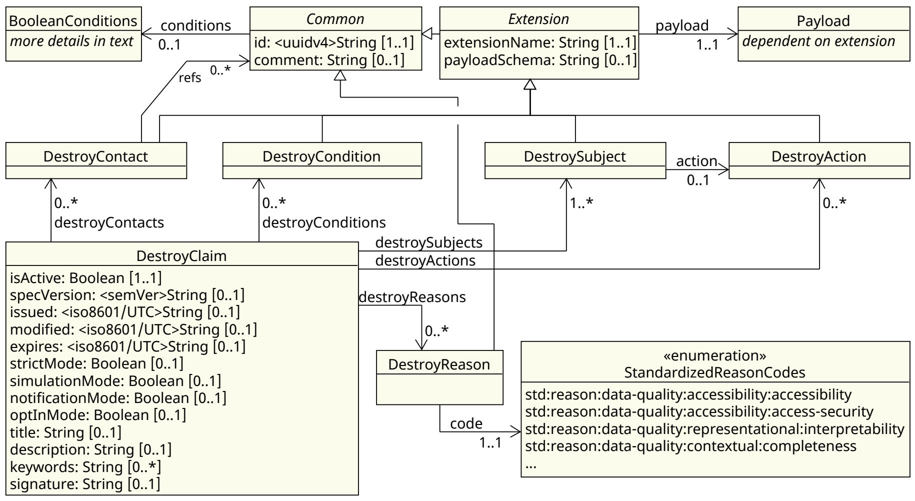

BooleanConditions
more details in text
conditions
0..1
Common
id: <uuidv4>String [1..1]
comment: String [0..1]
Extension
extensionName: String [1..1]
payloadSchema: String [0..1]
payload
1..1
Payload
dependent on extension
refs
0..*
DestroyContact
DestroyCondition
DestroySubject
action
0..1
DestroyAction
0..*
destroyContacts
0..*
destroyConditions
1..*
0..*
DestroyClaim
isActive: Boolean [1..1]
specVersion: <semVer>String [0..1]
issued: <iso8601/UTC>String [0..1]
modified: <iso8601/UTC>String [0..1]
expires: <iso8601/UTC>String [0..1]
strictMode: Boolean [0..1]
simulationMode: Boolean [0..1]
notificationMode: Boolean [0..1]
optInMode: Boolean [0..1]
title: String [0..1]
description: String [0..1]
keywords: String [0..*]
signature: String [0..1]
destroySubjects
destroyActions
destroyReasons
0..*
DestroyReason
code
1..1
«enumeration»
StandardizedReasonCodes
std:reason:data-quality:accessibility:accessibility
std:reason:data-quality:accessibility:access-security
std:reason:data-quality:representational:interpretability
std:reason:data-quality:contextual:completeness
...

Figure 2. Destroy Claim Model

Table 1. Destroy Claim Properties (descriptive)

Property	*Description*
id	The unique id to identify a Destroy Claim.
title	An easy to understand title that describe what the Destroy Claim is about.
description	A detailed description of the Destroy Claim.
keywords	Keywords which tag the Destroy Claim.
issued	Date of formal issuance of the Destroy Claim.
modified	Date on which the Destroy Claim was last modified.
comment	A human readable comment.
signature	A cryptographic signature to establish trust in the Destroy Claim.

that it is not possible to develop a single rigid model to describe the end of the data life cycle. For this reason, the aspects of *what*, *when*, *where*, *who*, and *how* can be replaced by individual solutions. An extension should have a unique *extensionName* to distinguish between them. Each type of extension can define its individual *payload*, in which the modeling power required for the use case is given. For an extension, it must be clearly defined when it evaluates positively and when it evaluates negatively. There are four concrete variants of an extension. *DestroyContact* is used to model *who* has responsibilities. The extension can indicate responsibilities using *refs* that link to other parts of the Destroy Claim. *DestroySubject* is used to address the data to be deleted. The addressing should be as precise as possible so that incorrect data is not selected and deleted. Here, for example, content hashes or UUIDs can be used to address data. Under *DestroyCondition*, the conditions can be modeled when the end of the data life cycle is reached. Here, aspects of *when*, *where* and *who* are to be modeled. For example, one could formulate three conditions stating that a certain file must be deleted when the year

Table 2. Destroy Claim Properties (interpretive)

Property	*Description*
isActive	Indicates if a Destroy Claim is active. If isActive is `false`, the Destroy Claim MUST NOT be executed.
specVersion	Indicates which version of the Destroy Claim Model Specification was used to generate this Destroy Claim (currently only 1.0.0 available).
expires	As of the specified date, the Destroy Claim MUST NOT be executed and can be deleted.
strictMode	If DCA (see Section 3.2) should act in strict mode or normal mode (for this and all other modes see GitHub (Tebernum 2023)).
simulationMode	If DCA should act in simulation mode or real mode.
notificationMode	If DCA should act in notification mode or silent mode.
optInMode	If DCA should ask for permission or can execute the Destroy Claim on its own.

2024 is reached, one is outside of Germany and one belongs to a certain department. *DestroyAction* is used to describe *how* data should be deleted. Here one can model, for example, whether memory areas are to be overwritten several times or whether the hardware must be destroyed.

A great promise of a unified model is that different parties can interpret the model in the same way. The use of *Extension* makes this goal no longer achievable without further ado. Therefore, we have designed a standard library of extensions to promote a common understanding and increase the chances that Destroy Claims can be understood outside one's own technical or organizational boundaries.[3]

Finally, we increased the flexibility of the Destroy Claims by using Boolean conditions. By default, all extension evaluations are combined using a Boolean AND. Each extension, the *DestroyClaim* class, and the *DestroyReason* class can use Boolean algebra to influence a positive or negative evaluation outcome. For this purpose, the evaluation results of other extensions can be used and linked with the help of Boolean operators. E.g., one can model that a Destroy Subject S_1 may only be deleted if another Destroy Subject S_2 is also deleted and only one of two additional conditions is met $((S_1 \Rightarrow S_2) \wedge (C_1 \oplus C_2))$.

3.2 An Architectural Integration Proposal (A2)

This artifact deals with the question of how to integrate A1 into software and thus support end-of-life data management. This artifact contributes to RQ2.

First, we need a trusted source for managing Destroy Claims. A data catalog is software that inventories all kinds of data by storing corresponding metadata. We suggest modelling the end of the data life cycle in a data catalog or similar.

Second, we designed the concept of the so-called Destroy Claim Agents (DCAs). The DCA is a software component that receives and interprets Destroy Claims from one or more trusted sources and executes the end of the data life cycle in the environment.

[3] `https://github.com/DaTebe/destroyclaims/blob/9a8a6e5e432312175cc439f333c45620924400fc/docs/std-extensions.md`

A DCA has two required interfaces. The *IDestroyClaim* interface allows the DCA to connect to one or more trusted sources for receiving Destroy Claims. The interface should be implemented according to the use case and can, for example, implement filters to query only relevant Destroy Claims or decide between a pull or push strategy. The *IEnvironment* interface connects the DCA to its environment. The DCA has two duties that it must implement. First, the DCA must monitor the environment to determine whether Destroy Claims are applicable. Second, the DCA must be able to interact with the persistence layer of the environment to be able to delete data. The DCA also has one optional provided interface *IControl* that e.g., a human actor can use to consent to the execution of Destroy Claims or to configure the environment to be monitored. A detailed description of how a DCA has to evaluate and execute Destroy Claims is available on GitHub.[4]

Third, we want to provide an overview of how Destroy Claims and DCAs can be integrated into software architectures. Our four integration proposals can be seen in Figure 3. In the following we will briefly characterize them.

central: Using this integration strategy, there is only one central DCA. This strategy is best used when one does not have the capacity to manage multiple DCAs and when one DCA is sufficient to manage a potentially heterogeneous data landscape. In the best case, you do not have to take care of the deployment of the DCA, but it may already be part of a data catalog or other trusted sources for Destroy Claims. A central DCA component may not gain an overview of the entire data landscape, because either the technologies in use are not supported or they are not accessible from a central point. In addition, the DCA must be configured to explicitly monitor and control certain systems.

stand alone: For most parts, this integration strategy is the same as *central*. The difference is that several individual DCAs divide up the data landscape. To avoid unwanted interference between DCAs, they should (if possible) strictly separate the data landscape between them. This integration strategy should be applied when the various environments are in different contexts, e.g. have distinct responsibilities due to corporate structures. The operation of several DCAs will lead to an increased amount of administrative work.

fully integrated: In this integration strategy, the DCA becomes an integral part of the service. This has the advantage that one no longer has to explicitly take care of the DCA. It only needs to be configured from which trusted source the Destroy Claims are to be obtained, when *IDestroyClaims* is implemented using a pull strategy. In addition, the DCA will be very much aligned with the existing data and technologies, which will likely make it more robust. This type of integration is suitable when a service is implemented from scratch. The costs would otherwise probably exceed the benefits. One disadvantage may be that it may not be able to be customised to interpret further extensions, but that you are dependent on the vendor of the specific service.

side car: This integration strategy is similar to the *fully integrated* one. The DCA is not part of the service here, but lives alongside it. However, it is completely dependent on the life cycle of the service. If the service is stopped, the DCA is also terminated. The example in Figure 3 shows a DCA inside an application that monitors the file system

[4] https://github.com/DaTebe/destroyclaims/blob/9a8a6e5e432312175cc439f333c45620924400fc/docs/destroy-claim.md#dca-evaluating-and-executing-a-destroy-claim

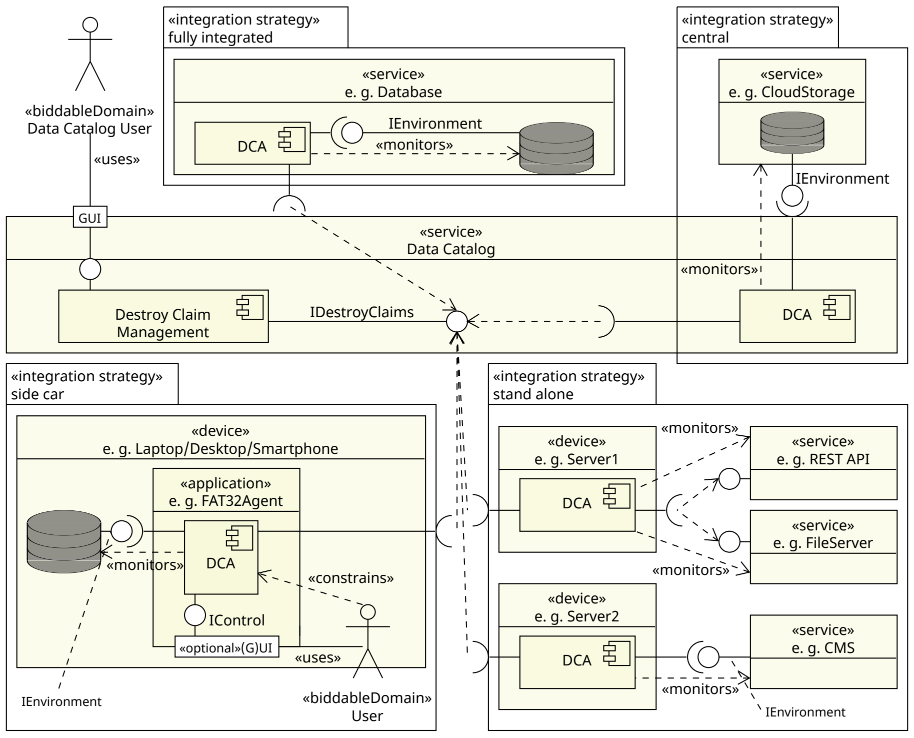


Figure 3. Architectural Proposals for Destroy Claim and DCA Integration

of an operating system. When the operating system and its file system is shut down, the DCA is also stopped. This strategy has the advantage that it is easy to deploy and scale alongside the service. It should be noted that the management of many DCAs can be difficult and resources are bound more quickly than, for example, with the central strategy.

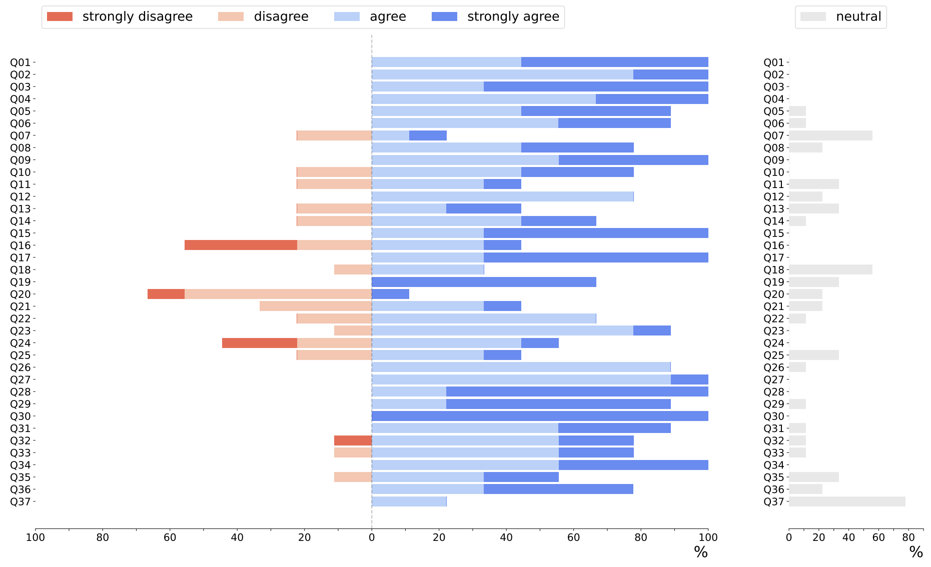
4 Demonstration

The demonstration is performed using implemented prototypes of the artifacts. We extended the open source data catalog DIVA to support Destroy Claims (Tebernum & Atamantschuk 2023). Here, our concrete derivation of the model using JSON is implemented. Users can utilize the *DestroyClaimManagement* component to create and configure Destroy Claims via a web interface. Furthermore, the *IDestroyClaims* interface was implemented that enables external applications to retrieve Destroy Claims using a pull strategy. We also implemented a DCA with the *destroyclaim-js* library, which can also be found in our Destroy Claim GitHub repository (Tebernum 2023).

The use case itself deals with modeling the end of the data life cycle of file-based data. It was derived from real cases in the business world, where employees work with e.g. *Word*, *PowerPoint*, or *Photoshop*. Files are copied from different systems, such as the company's network or the internet, to one's local device. If the data is no longer to be used for various reasons, e.g. because it is out of date or has errors, employees may not be aware of this. They are currently encouraged to detect and remove obsolete data by themselves. We have set up a system that implements the sidecar integration strategy (see Figure 3). We used DIVA as our trusted source and to inventory the file-based data. Users can model a Destroy Claim when they inventory data in DIVA. A DCA with a graphical interface was implemented that creates an index of the locally available data on the devices of the employees. When the end of the data life cycle is reached, the employees are informed via a pop-up. If the Destroy Claim is running in *optInMode*, the user can still decide whether to agree to the claim.

5 Evaluation

During the focus groups, several topics emerged that should be considered in future artifact iterations. An important point was that Destroy Claims are only useful if they are mostly automatically generated. It was suggested that there must be standardized blueprints that can be applied to classes or types of data. Data catalogs were identified as a useful component to perform this automation. The experts suggested that the Destroy Claims themselves must also be deleted at some point and whether there are Destroy Claims for Destroy Claims. We address this by setting an expiration date after which a Destroy Claim is no longer valid. Destroy Claims that are valid forever, for example because they address illegal data, would in fact theoretically exist forever. The general consensus was that this could probably only be a problem on a global scale where one has no control over the environment. Another point that was addressed was that a deletion, that is dependent on another deletion, is only possible if a single DCA monitors the environments in which the two data sets occur. There is no provision for multiple DCAs to collaborate. The experts argue that some data deletion could never be executed because of this. Another limitation concerns the intervals at which a Destroy Claim should be tested, as their applicability can change quickly depending on the conditions set. Strategies for regular evaluation of Destroy Claims need to be developed, and experts have suggested that evaluation strategies or claim priority could be included in the model's next iteration. While explaining the Detroy Claim model, two participants highlighted the similarity of the concrete JSON implementation to ODRL (Iannella & Villata 2018), and it should be considered whether a concrete instantiation of the model and the transfer of standard extensions to the ODRL world are feasible and beneficial. In addition to the gaps addressed, the experts also identified use cases that the authors had not considered. One idea was to have a global system, maybe based on the blockchain technology, in which Destroy Claims could be collected. Creators of intellectual property can e.g., file claims for pirated copies. Another expert imagined placing signatures of viruses, spam and the like there. One question that came up here was about trustworthiness. The experts agreed that with digital signatures and trusted publishers, it could work. The experts also pointed out that even if the model is not used one-to-one in practice, it is a good basis for deriving more specialized profiles.



General:
Q01 Modelling the end of the data life cycle is important. **Q02** Destroy Claims have the potential to professionalize the end of the data life cycle. **Q03** A data catalog is the right place to model the end of the data life cycle.

Goal-based: (A1) The Destroy Claim Model...
Q04 ... is generally/in theory suitable/relevant to model the end of the data life cycle. **Q05** ... is suitable/relevant for modelling the end of the data life cycle in practice. **Q06** ... correctly achieves its goal. **Q07** ... has a return on investment that justifies the implementation costs. **Q08** ... is generally valid/applicable in the topic of modeling the end of the data life cycle. **Q09** ... can be technically implemented. **Q10** ... can be used in the context of existing technologies. **Q11** ... is economically reasonable.

Structure-based: (A1) The Destroy Claim Model...
Q12 ... is complete. **Q13** ... follows the KISS (Keep it small, stupid!) principle. **Q14** ... is simple to understand. **Q15** ... is consistent.

Environment-based: (A1) The Destroy Claim Model...
Q16 ... is something I would personally use/ would be useful to me. **Q17** ... is useful for data engineers. **Q18** ... can be easily deployed/used. **Q19** ... is ethically justifiable. **Q20** ... has no side effects. **Q21** ... converges with the business goals of my company. **Q22** ... has no negative impact on (my) business. **Q23** ... can be integrated into existing technologies. **Q24** ... uses innovative technologies. **Q25** ... has no negative impact on the technology in use.

Evolution-based: (A1) The Destroy Claim Model...
Q26 ... is robust enough to withstand increasing stress. **Q27** ... can be scaled (e.g. to work globally). **Q28** ... can be adapted into other contexts. **Q29** ... can be modified for future challenges.

Activity-based: (A2) The implementation and integration strategies for Destroy Claims...
Q30 ... increase the awareness to model the end of the data life cycle. **Q31** ... increase the acceptance to model the end of the data life cycle. **Q32** ... increase trust in the data. **Q33** ... increase confidence in deleting data. **Q34** ... can (theoretically) cover my personal data deletion use cases. **Q35** ... correspond to the KISS (Keep it small, stupid!) principle. **Q36** ... are consistent. **Q37** ... are performant

Figure 4. Survey Results and Questions

In order to get systematic feedback, the experts were given a survey after the focus group sessions (see Figure 4). To provide an initial overview of the success of the project, some general questions were asked (Q01-Q03). It can be noted that the responses from the participants were consistently positive. Therefore, we can be confident that we are on the right track in making the end of the data life cycle a first-class citizen in data engineering. Following, a series of questions is asked to check whether artifact A1 achieves its objectives (Q04-Q11). Here, too, the assessment is predominantly positive. However, it is striking that Q07 is predominantly answered neutrally. The participants don't seem to have any experience or assessment as to whether it is financially worthwhile to implement the solution. It is generally difficult to obtain reliable statistics in this regard. Nevertheless, it would be interesting to look at this aspect more closely in the future. The next block deals with the structure of A1 (Q12-Q15). This, too, is predominantly perceived positively. In detail, future iterations should look at whether A1 can be made simpler (Q13), as it can become objectivly quite complex to create and interprete Destroy Claims. Next, there is a series of questions dealing with the effect of A1 on the environment (Q16-Q25). There are a few points to discuss here. First of all, it is noticeable that most of the participants would not use the artefact A1 themselves (Q16). We suspect that the solution creates far too much overhead for individuals and is not a good fit in this context. This is supported by the fact that the artifact is considered useful for data engineers, who create enterprise-wide solutions and work in bigger environments (Q17). When we look at Q18, we have many neutral reports when it comes to ease of use and deployment. This is a good sign that the respondents answered mindfully, as they do not have any empirical data in this regard. The question whether the solutions have side effects on the environment (Q20), are negatively evaluated. We believe, this reflects the fear that deleting data will lead to unwanted states in which e.g., dead references are created or parts of the whole system no longer work. In the future, a clear focus in research should be on how to minimize side effects, by e.g., looking into formal guarantees or at least well-tested implementation patterns. The next set of questions pertains to the adaptability of A1 to future challenges (Q26-Q29). Again, the responses were consistently positive. This can further support the practical implementation of A1, as the necessary flexibility is present in case of required adjustments. Finally, questions were asked whether artifact A2 is achieving its objectives (Q30-Q37). Overall, the responses here are also very positive. Noteworthy is Q30, where all participants have consistently responded with *strongly agree*. We see this as an indicator that A2 makes a valuable contribution in supporting end-of-life data management. Also, the question about the performance of A2 (Q37) was predominantly answered neutrally. Again, this shows that the participants filled out the questionnaire mindfully, as they could not have any information in this regard.

6 Related Work

The end of the data life cycle is a phase that is not considered in many data life cycle models ranging from research (Lenhardt et al. 2014, Kowalczyk 2017, Wissik & Ďurčo 2016, Ma et al. 2014, Faundeen et al. 2013, Allard 2012, Patel 2016), metadata (Catteau et al. 2006, Kosch et al. 2005), agriculture (Demestichas & Daskalakis 2020) to big data (Khan et al. 2014, Demchenko et al. 2014) and machine learning (Miao et al. 2017). If it is mentioned (Yu & Wen 2010, Michota & Katsikas 2015, Lin et al. 2014, Chaki & Chaki 2015, Hubert Ofner et al. 2013), it is only in passing and without reference to further literature. Likewise, data management frameworks that investigate the end of the data life cycle provide, at best, general guidelines for deleting data (Shah et al. 2021). In practice, technical solutions are found that feature end-of-life data management. For example, in the context of data life cycle policies in proprietary software such as Microsoft Azure (2023), Google Cloud (2023*b*) or in the form of usage policies in ODRL (2018). These solutions have in common that they do not consider the end of the data life cycle in more detail, but describe it as a by-product in the form of yet another action on data to be performed under certain conditions. To the best of our knowledge, there is no work in research that deals with modelling the end of the data life cycle or its management in a central and comprehensive way or addresses it in the context of data engineering.

7 Conclusion

Despite the fact that the end of the data life cycle and its management are becoming increasingly important, there is little to no attention in literature and practice. Our expert interviews indicate, that this is due to a missing common ground that creates a shared understanding among data engineering experts and practitioners. We address this by providing two DSR artifacts. First, Destroy Claims which comprehensively model the end of the data life cycle. Second, an architectual proposal that describes how end-of-life data management can be integrated into software utilizing Destroy Claims. The technical feasibility was demonstrated using prototypes that were implemented and made publicly available. Our evaluation found that the artifacts were mostly positively received and adequately addresse the objectives.

While Chapter 5 has already addressed limitations and future work related to the artifacts, this section will focus on methodological constraints. Firstly, it is important to acknowledge the potential for selection bias given the small sample size of six interviews and nine participants in the evaluation. Additionally, the coding process was conducted solely by the authors, raising the possibility that the derived objectives may not accurately represent the broader population of data engineers. Lastly, in terms of methodological considerations and future research, it is essential to deploy the artifacts in real-world scenarios to better assess their usefulness. However, this undertaking requires substantial effort and costs, as the artifacts intervene very fundamentally in existing software systems. Therefore, we will initially conduct smaller-scale studies before undertaking larger-scale implementations.

Acknowledgments

The authors acknowledge the financial support by the Federal Ministry for Economic Affairs and Climate Action of Germany in the project Smart Design and Construction (project number 01MK20016F).

References

Allard, S. (2012), ‘Dataone: Facilitating escience through collaboration’, *Journal of eScience Librarianship* **1**(1).

Catteau, O., Vidal, P. & Broisin, J. (2006), A generic representation allowing for expression of learning object and metadata lifecycle, *in* ‘Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies, ICALT 2006, Kerkrade, The Netherlands, July 5-7, 2006’, IEEE Computer Society, pp. 30–32.
URL: *https://doi.org/10.1109/ICALT.2006.1652357*

Chae, B. K., Yang, C., Olson, D. & Sheu, C. (2014), ‘The impact of advanced analytics and data accuracy on operational performance: A contingent resource based theory (rbt) perspective’, *Decision support systems* **59**, 119–126.

Chaki, S. & Chaki, S. (2015), ‘The lifecycle of enterprise information management’, *Enterprise Information Management in Practice: Managing Data and Leveraging Profits in Today’s Complex Business Environment* pp. 7–14.

Corbin, J. M. & Strauss, A. (1990), ‘Grounded theory research: Procedures, canons, and evaluative criteria’, *Qualitative sociology* **13**(1), 3–21.

Council of the EU, G. S. o. t. C. (2022), ‘Energy crisis: Three eu-coordinated measures to cut down bills’.
URL: *https://www.consilium.europa.eu/en/infographics/eu-measures-to-cut-down-energy-bills/*

Demchenko, Y., De Laat, C. & Membrey, P. (2014), Defining architecture components of the big data ecosystem, *in* ‘2014 International conference on collaboration technologies and systems (CTS)’, IEEE, pp. 104–112.

Demestichas, K. & Daskalakis, E. (2020), ‘Data lifecycle management in precision agriculture supported by information and communication technology’, *Agronomy* **10**(11), 1648.

European Commission (2016), ‘Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)’.
URL: *https://eur-lex.europa.eu/eli/reg/2016/679/oj*

Faundeen, J. L., Burley, T. E., Carlino, J., Govoni, D. L., Henkel, H. S., Holl, S., Hutchison, V. B., Martín, E., Montgomery, E. T., Ladino, C. C. et al. (2013), *The United States geological survey science data lifecycle model*, US Department of the Interior, US Geological Survey Reston, VA, USA.

Gao, B., Chen, B., Jia, S. & Xia, L. (2019), ehifs: An efficient history independent file system, *in* ‘Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security’, pp. 573–585.

Glaser, B. G., Strauss, A. L. & Strutzel, E. (1968), ‘The discovery of grounded theory; strategies for qualitative research’, *Nursing research* **17**(4), 364.

Google Cloud (2023*a*), ‘Announcement of pricing changes for cloud storage’, `https://cloud.google.com/storage/pricing-announce`. [Online; accessed 6-June-2023].

Google Cloud (2023*b*), ‘Object lifecycle management’, `https://cloud.google.com/storage/docs/lifecycle`. [Online; accessed 7-June-2023].

Hollander, J. A. (2004), ‘The social contexts of focus groups’, *Journal of contemporary ethnography* **33**(5), 602–637.

Hubert Ofner, M., Straub, K., Otto, B. & Oesterle, H. (2013), ‘Management of the master data lifecycle: a framework for analysis’, *Journal of Enterprise Information Management* **26**(4), 472–491.

Iannella, R. & Villata, S. (2018), ‘ODRL Information Model 2.2’.
URL: *https://www.w3.org/TR/odrl-model/*

Jackson, T. W. & Hodgkinson, I. R. (2022), ‘Keeping a lower profile: how firms can reduce their digital carbon footprints’, *Journal of Business Strategy* (ahead-of-print).

Jung, C., Dörr, J., Otto, B., Ten Hompel, M. & Wrobel, S. (2022), ‘Data usage control’, *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage* pp. 129–146.

Khan, N., Yaqoob, I., Hashem, I. A. T., Inayat, Z., Mahmoud Ali, W. K., Alam, M., Shiraz, M. & Gani, A. (2014), ‘Big data: survey, technologies, opportunities, and challenges’, *The scientific world journal* **2014**.

Klonowski, M., Struminski, T. & Sulkowska, M. (2019), Universal encoding for provably irreversible data erasing., *in* ‘ICETE (2)’, pp. 137–148.

Kosch, H., Böszörményi, L., Döller, M., Libsie, M., Schojer, P. & Kofler, A. (2005), ‘The life cycle of multimedia metadata’, *IEEE Multim.* **12**(1), 80–86.
URL: *https://doi.org/10.1109/MMUL.2005.13*

Kowalczyk, S. T. (2017), ‘Modelling the research data lifecycle’, *Int. J. Digit. Curation* **12**(2), 331–361.
URL: *https://doi.org/10.2218/ijdc.v12i2.429*

Lenhardt, W., Ahalt, S., Blanton, B., Christopherson, L. & Idaszak, R. (2014), ‘Data management lifecycle and software lifecycle management in the context of conducting science’, *Journal of Open Research Software* **2**(1).

Lin, C.-W., Hong, T.-P. & Lu, W.-H. (2009), An efficient fusp-tree update algorithm for deleted data in customer sequences, *in* ‘2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)’, IEEE, pp. 1491–1494.

Lin, L., Liu, T., Hu, J. & Zhang, J. (2014), A privacy-aware cloud service selection method toward data life-cycle, *in* ‘2014 20th IEEE international conference on parallel and distributed systems (ICPADS)’, IEEE, pp. 752–759.

Ma, X., Fox, P., Rozell, E., West, P. & Zednik, S. (2014), ‘Ontology dynamics in a data life cycle: challenges and recommendations from a geoscience perspective’, *Journal of Earth Science* **25**, 407–412.

Miao, H., Li, A., Davis, L. S. & Deshpande, A. (2017), Towards unified data and lifecycle management for deep learning, *in* ‘33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017’, IEEE Computer Society, pp. 571–582.
URL: *https://doi.org/10.1109/ICDE.2017.112*

Michota, A. & Katsikas, S. (2015), Designing a seamless privacy policy for social networks, *in* ‘Proceedings of the 19th panhellenic conference on informatics’, pp. 139–143.

Microsoft Azure (2023), ‘Optimize costs by automatically managing the data lifecycle’, `https://learn.microsoft.com/en-us/azure/storage/blobs/lifecycle-management-overview`. [Online; accessed 7-June-2023].

ODRL (2018), ‘Odrl information model 2.2’, `https://www.w3.org/TR/odrl-model/`. [Online; accessed 7-June-2023].

Österle, H., Winter, R. & Brenner, W. (2010), *Gestaltungsorientierte Wirtschaftsinformatik: Ein Plädoyer für Rigor und Relevanz*, Infowerk.

Othon, M., Mile, S., de Melo, A., Junior, D. A. & Arruda, A. (2019), ‘Evaluation of the removal of anomalies in data collected by sensors’.

Otto, B. (2015), ‘Quality and value of the data resource in large enterprises’, *Information Systems Management* **32**(3), 234–251.

Pachpor, N. N. & Prasad, P. S. (2018), Improving the performance of system in cloud by using selective deduplication, *in* ‘2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)’, IEEE, pp. 314–318.

Patel, D. (2016), ‘Research data management: a conceptual framework’, *Library review* **65**(4/5), 226–241.

Peffers, K., Tuunanen, T., Rothenberger, M. A. & Chatterjee, S. (2007), ‘A design science research methodology for information systems research’, *Journal of management information systems* **24**(3), 45–77.

Prat, N., Comyn-Wattiau, I. & Akoka, J. (2015), ‘A taxonomy of evaluation methods for information systems artifacts’, *Journal of Management Information Systems* **32**(3), 229–267.

Reardon, J., Capkun, S. & Basin, D. (2012), Data node encrypted file system: Efficient secure deletion for flash memory, *in* ‘21st USENIX Security Symposium (USENIX Security 12)’, pp. 333–348.

Shah, S. I. H., Peristeras, V. & Magnisalis, I. (2021), ‘Dalif: a data lifecycle framework for data-driven governments’, *J. Big Data* **8**(1), 89.
URL: *https://doi.org/10.1186/s40537-021-00481-3*

Tallon, P. P., Ramirez, R. V. & Short, J. E. (2013), ‘The information artifact in it governance: toward a theory of information governance’, *Journal of Management Information Systems* **30**(3), 141–178.

Tebernum, D. (2023), ‘Destroy Claims’. DOI: 10.5281/zenodo.8026996.
URL: *https://github.com/DaTebe/destroyclaims*

Tebernum, D., Altendeitering, M. & Howar, F. (2021), DERM: A reference model for data engineering, *in* C. Quix, S. Hammoudi & W. M. P. van der Aalst, eds, ‘Proceedings of the 10th International Conference on Data Science, Technology and Applications, DATA 2021, Online Streaming, July 6-8, 2021’, SCITEPRESS, pp. 165–175.
URL: *https://doi.org/10.5220/0010517301650175*

Tebernum, D., Altendeitering, M. & Howar, F. (2023), A survey-based evaluation of the data engineering maturity in practice. In press by Springer (release 2023) (see `https://www.researchgate.net/publication/367309981_A_Survey-based_Evaluation_of_the_Data_Engineering_Maturity_in_Practice`).

Tebernum, D. & Atamantschuk, S. (2023), ‘DIVA - Data Inventory and Valuation Approach’.
URL: *https://github.com/FraunhoferISST/diva*

Tebernum, D. & Howar, F. (2023), Structuring the end of the data life cycle. In press by INSTICC (release 2023) (see `https://www.researchgate.net/publication/371499692_Structuring_the_End_of_the_Data_Life_Cycle`).

Trajanov, D., Zdraveski, V., Stojanov, R. & Kocarev, L. (2018), Dark data in internet of things (iot): challenges and opportunities, *in* '7th Small Systems Simulation Symposium', pp. 1–8.

Venable, J., Pries-Heje, J. & Baskerville, R. (2012), A comprehensive framework for evaluation in design science research, Vol. 7286, pp. 423–438.

Wissik, T. & Ďurčo, M. (2016), Research data workflows: from research data lifecycle models to institutional solutions, *in* 'Selected papers from the CLARIN annual conference 2015, October 14–16, 2015, Wroclaw, Poland', number 123, Linköping University Electronic Press, pp. 94–107.

Yu, X. & Wen, Q. (2010), A view about cloud data security from data life cycle, *in* '2010 international conference on computational intelligence and software engineering', IEEE, pp. 1–4.