Extending the design space of ontologization practices: Using bCLEARer as an example

Chris Partridge
BORO Solutions Ltd
University of Westminster
London, UK
0000-0003-2631-1627

Andrew Mitchell
BORO Solutions Ltd
University of Westminster
London, UK
0000-0001-9131-722X

Sergio de Cesare
Westminster Business School
University of Westminster
London, UK
0000-0002-2559-0567

John Beverley
Department of Philosophy
University at Buffalo
Buffalo, USA
0000-0002-1118-1738

***Abstract*— Our aim in this paper is to suggest that the possible design space for the ontologization process is richer than current practice would suggest. That it is possible to open the space up to a range of radically new practices. This consciously builds upon the notion that engineering processes as well as products need to be designed. We provide evidence for the new practices from our work over the last three decades with an outlier methodology, bCLEARer. We also provide some contextual scaffolding for a perspective that we have found we needed to better understand the nature of these new practices. This is an evolutionary perspective which sees digitalization (the evolutionary emergence of computing technologies) as part of the latest step in a long evolutionary trail of information transitions. And sees ontologization as a tool for exploiting the emerging opportunities offered by digitalization.**

Keywords—information transmission, information evolution, bCLEARer methodology, ontologization methodologies, ontologization design space, computerization, digitization

I. INTRODUCTION

In ontology engineering there is, in theory at the very least, a tight coupling between ontologies and ontologization, the process that produces them. Our aim in this paper is to suggest that the possible design space for the ontologization process is richer than a look at many of the current methodologies would suggest. To suggest that it is possible to open the space up to a range of radically new practices.

In large part, the evidence for this is derived from our work over the last three decades with an outlier methodology, bCLEARer. We think it provides a useful example of some of these practices. We hope that by outlining these practices here we will make the case for expanding the design space and encourage the community to adopt a wider range of practices.

We also believe that ontologization needs some contextual scaffolding to provide the perspective needed to better understand the nature of these new practices. That one should look at it from an evolutionary perspective and see it as part of the latest step in a long evolutionary trail of information transitions – digitalization (the evolutionary emergence of computing technologies). Where ontologization is then revealed as a tool for exploiting the emerging opportunities offered by digitalization.

A. Structure of the paper

In the next, second, section, we provide a broad picture of the ontologization process and introduce two current ontologization methodologies to act as a baseline for comparisons. In the third section we briefly introduce our example, the bCLEARer process. In the fourth section we build the contextual scaffolding, firstly situating digitalization into an evolutionary perspective, then situation ontologization within digitization. In the fifth section, we situation bCLEARer into this evolutionary perspective. In the sixth section, we illustrate from within the evolutionary perspective some of the outlier design choices that bCLEARer has made.

II. A BROAD PICTURE OF THE ONTOLOGIZATION PROCESS

In ontology engineering there is, in theory at the very least, a tight coupling between ontologies and ontologization, the process that produces them. One can characterize this as a product-process relationship. The product-process distinction is a fundamental concept in traditional engineering. It differentiates the product (the output) and the process (the method or system used to produce the output). And the engineering mindset expects both the product and the process to be engineered. And as part of the engineering to both be quality managed, hence quality assurance (process) and quality control (product).

A. Top-level ontologies and ontologization

In top-level ontology (engineering) work, the process is often an invisible relation of the product. An example of this is provided by the main standard, ISO 21838-1:2021 - Information technology: Top-level ontologies (TLO) – Requirements. This discusses the ontology of processes but makes no mention of the ontologization process. Hence, unsurprisingly, the standards based upon it do not mention the ontologization process either.

Some of the top-level ontologies have support for the ontologization process. The top-level Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) has a related analysis tool OntoClean [1], but this falls short of an ontologization process. The Basic Formal Ontology (BFO) has a book [2] on the ontologization process that assumes the BFO top-level ontology. We look at this text in more detail below. The BORO Foundational Ontology has a closely intertwined bCLEARer ontologization process [3] [4]. The bCLEARer

ontologization process assumes a minimal top-level ontology to guide the work.

There are a variety of domain level ontologization processes that we discuss later in this paper.

B. The case for engineering the ontologization process

The importance of engineering the process is reflected in the often-quoted dictum that: the quality of the process determines the quality of the product.

For a historical background to this, from the wider history of innovation, see Mokyr's *The Past and the Future of Innovation* [5] or his *A Culture of Growth* [6]. He argues that history shows that technological progress cannot rely on artisanal skills alone, it needs to be supplemented with formal and systematic (that is, engineered) knowledge.

Within engineering, this idea was explored, analyzed and championed in manufacturing in the second half of the 20th century by quality management pioneers such as W. Edwards Deming [7] and Joseph Juran [8]. This led to a rich variety of designs including movements such as Total Quality Management and its successors Lean Manufacturing, and Six Sigma. These developed a rich range of ways of managing manufacturing processes. An example is the Plan-Do-Check-Act (PDCA) Cycle used to design, implement, and refine processes on a small scale – which fits well with the Kaizen philosophy of continuous improvement, where processes are regularly reviewed and improved incrementally. This has spread to some other domains. For example, one can see Kaizen-like principles being used in the Agile software development methodology.

Within the ontology engineering community, one does not find a comparatively rich selection of designs and range of ways of managing the ontologization processes – a kind of process design poverty. Especially for top-level ontologies, there appears to be more 'theory' for, and so more attention on, the design of the final product – 'the 'ontology' – than the process – 'ontologization' – that produces it. From an engineering perspective, this imbalance looks unhealthy.

This imbalance is unlikely to be by choice as the benefits of designing the process are obvious. The most obvious is that in the long term, the savings from investment in quality far outweigh the upfront cost of development. One could argue that this poverty arises from the process being relatively new and under-researched, unlike, for example, top-level ontology which can build upon a rich heritage.

1) Process design poverty in logic

Interestingly, a similar poverty of design process has been pointed out in logic, which is a key part of the last stages of the 'ontologization' process. Novaes [9] makes a product-process distinction for logic: distinguishing the formal product from its formalization process:

"As a discipline, logic is arguably constituted of two main sub-projects: formal theories of argument validity on the basis of a small number of patterns, and theories of how to reduce the multiplicity of arguments in non-logical, informal contexts to the small number of patterns whose validity is systematically studied (i.e. theories of formalization). Regrettably, we now tend to view logic 'proper' exclusively as what falls under the first sub-project, to the neglect of the second, equally important sub-project."

She discusses two historical theories of argument formalization, from Aristotle and Medieval Logic. Both "illustrate this two-fold nature of logic, containing in particular illuminating reflections on how to formalize arguments (i.e. the second sub-project)." She suggests reflecting on these should lead to a broader conceptualization of what it means to formalize.

Given how much the ontology (engineering) product builds on formal logic – inheriting many of its (cultural) practices – this may contribute to the poverty in ontology engineering. This suggests that developments in the formalization process could be recruited by and enrich ontologization's approach to formalization.

C. Comparing different ontologization processes

In this section, we briefly look at some current mainstream methodologies that guide the ontologization process to provide a basis for comparison with bCLEARer. This gives us a rough benchmark on common practices. A quick caveat: we are not claiming that this selection reflects all the work that is happening in this area. Rather, we are aiming for cases that lend themselves to our broad comparison.

As we touch upon later, from a bCLEARer perspective, there are interesting features in the methodologies guiding the processes in other software related domains, such as:

- Waterfall model: clear top-down separation of concerns.
- Agile: flexible, responsive, efficient, iteration
- DevOps (and DataOps): automation into a data pipeline to improve and shorten life cycles.

In this section we restrict ourselves to ontologization to help make a clear comparison. We roughly divide our selected domain of ontologization approaches into two broad camps, which we have colloquially labelled: 'ask-an-expert' and 'top-down-classification'. There is a reasonably rich literature on ontology methodologies (KBSI. 1994), (Uschold. 1995), [13] , (Jones. 1998.) (Fernández-López. 2002), (Cristani. 2005), (Iqbal. 2013). From these we have selected [12] and [2] to represent the two camps.

Commented [CP1]: To be updated

One aspect of these methodologies we highlight is the information pathway they create, the flow or movement of information through the stages of the overall process. We specifically explore how this interacts with two levels of classification for information. Firstly, the level of generality which, for ease of understanding, we introduce from a data perspective as the metadata, schema and data levels. The is a simplification as it about the syntax of the implementation, whereas generality is a semantic matter. However, there is good enough rough match between syntax and semantics here to make the substitution fair. Secondly, the evolutionary levels of digitalization [10], [11], which we take here to be brain, speech, writing, printing and computing.

1) The 'ask-an-expert' approach

For simplicity, we take the OntoCommons report D.4.2 [12] as our basis, as it not only describes its own approach (the LOT methodology) but documents other similar approaches (Grüninger & Fox [13], METHONTOLOGY, On-To-Knowledge, DILIGENT, NeOn, RapidOWL, SAMOD and AMOD). Together these provide many good examples of the 'ask-an-expert' approach, which has its roots in Artificial Intelligence (AI) and knowledge representation.

The input for the process is domain experts – as this quote neatly illustrates:

"The goal of the ontology implementation activity is to build the ontology using a formal language, based on the ontological requirements identified by the domain experts." p. 28

Across all the approaches reviewed, there is a similar information pathway from a level of digitalization perspective. In the early stages there is an underlying focus on natural language, sometimes organized into (natural language) competency questions [13] – as this quote illustrates:

"If domain experts have no knowledge about ontology data generation and querying, we recommend writing the requirements in the form of natural language sentences." p. 22

The methodology's input to the information pathway is the brains of experts via speech into documented (unstructured) natural language. Then the methodology broadly separates concerns [14]: separating the confirmation of content from its formalization – and chooses to address the first concern before the second. A point we shall return to later is that this separation choice assumes there is no serious dependency between the content agreement and the formalization process. In this design architecture, the first stage is a confirmation of content which uses mostly (unstructured) natural language which is organized and agreed as a statement of the requirements of the ontology. The second stage takes the natural language and formalizes them.

The early pathway is not always or entirely natural language, as there is a mention of the possibility of using structured information in the shape of a "tabular technique" using "3 types of tables: Concepts, Relations, Attributes". Formalization (structured information) only really enters the process in the later stages of the pathway in ontology implementation, after the requirements (expressed in natural language) are collected.

"The goal of the ontology implementation activity is to build the ontology using a formal language, based on the ontological requirements identified by the domain experts" p. 28

The paper notes that there is optionally a conceptualization stage, where an interim concept model based upon the requirements may be built. Interestingly, it suggests that "*diagraming tools such as MS Visio or draw.io, as well as non-digital tools as pen and paper or a blackboard*" may be used to build this.

2) The 'top-down-classification' approach

We take Arp et al [2] as the baseline for this approach. This provides a good, clear example of the top-down-classification approach, with a clear summary of how they aim to construct an ontology (which shows why it deserves the top-down-classification nickname). This process is a rationalist armchair exercise that works top-down to set up the classification framework - rationalist in the sense that there is little empirical content. This has roots in biological classification and philosophy is a common approach to developing top-level ontologies in Information Systems (IS).

"Overview of the Domain Ontology Design Process Ontology is a top-down approach to the problem of electronically managing scientific information. This means that the ontologist begins with theoretical considerations of a very general nature on the basis of the assumption that keeping track of more specific information (for example, about specific organs, genes, or diseases) requires getting the very general scientific framework underlying this information right, and doing so in a systematic and coherent fashion. It is only when this has been done that the detailed terminological content of a specific science such as cell biology or immunology can be encoded in such a way as to ensure widespread accessibility and usability." [2, p. 49]

This informal view is then structured into a step-by-step process in a table – see below.

Table 3.1
An outline of the steps to be followed in designing a domain ontology
1. Demarcate the subject matter of the ontology.
2. Gather information: identify the general terms used in existing ontologies and in standard textbooks; analyze to remove redundancies.
3. Order these terms in a hierarchy of the more and less general ones.
4. Regiment the result in order to ensure:
a. logical, philosophical, and scientific coherence,
b. coherence and compatibility with neighboring ontologies, and
c. human understandability, especially through the formulation of human-readable definitions.
5. Formalize the regimented representational artifact in a computer usable language in such a way that the result can be implemented in some computable framework.
[2, p. 49]

From this table, we can pull out a level of digitalization perspective across the information pathway. The first four stages work with unstructured natural language. Though the terms 'order' and 'regiment', at steps 3 and 4, suggest some structure in the information, it is only at step 5 that the information is formalized, and so fully structured data enters the process. So here as well, the information pathway to the ontology starts with brains then via speech or directly into text is documented (unstructured) natural language.

In the process, there is a similar reliance upon human experts to justify choices, see:

"The terms in an ontology are the linguistic expressions used in the ontology to represent the world, and drawn as nearly as

possible from the standard terminologies used by human experts in the corresponding discipline." [2, p. 5]

As an aside, it is often not recognized that these terms are also elements of the domain that need to be represented in the ontology

3) Process Comparison

One can make a rough assessment of the maturity of these methodologies. As the quotes above hint at, they are currently collections of "ad hoc rules" with simple heuristics. There is no background context to act as a foundation to guide the engineering of the process design – certainly no common context. Hence, they are, from an engineering design perspective, at an early stage of development. There is still plenty of scope for them to undergo the kind of serious engineering re-design Deming and Juran undertook for manufacturing.

Both approaches have several features in common, ones that differentiate them from bCLEARer.

From the perspective of digitalization, in both cases, their early processes for establishing the base ontology focus their efforts on working with unstructured (pre-digitalization) natural language related to human understandability, with less focus on machine understandability. In both approaches the ontologization happens before the formalization. The ontological information in (unstructured) is captured and regimented in natural language first and then formalized.

One can broadly divide information into levels of generality. From a syntactic data perspective, these are the natural levels: metadata, schema and data. For our purposes here, these are a good enough rough proxy for the semantic top, most general, middle and the most specific bottom level. (We can't use the partially equivalent distinction in ontology between top-level (or upper) ontologies and domain ontologies as this has no term for the all-important data level.)

We can use this perspective to show how the two approaches showcased differ and agree. They differ in their approach to the metadata level. The 'top-down-classification' approach works top-down, starting with the metadata level and working down to the middle schema level. The 'ask-an-expert' approach works solely at the schema level – focusing on the domain. In principle, there is no reason why the 'ask-an-expert' approach could not start with the metadata level, or the 'top-down-classification' approach could not ignore the metadata level and work at the schema level. The two approaches agree on their approach to the bottom data level. They both ignore it (a significant omission, we return to below).

One can relate the design of the digitalization and generality features. If one chooses to design a process to work with humans and unstructured data, one needs to recognize that one is building in scaling constraints that block the processing of large amounts of data. One way around this is, of course, to work with data indirectly through the schema. bCLEARer takes a different route and aims to automate the process so removing these scaling constraints.

III. bCLEARer - History

In this section, we give a brief background history of bCLEARer and its place in BORO (an acronym for '*Business Object Reference Ontology*') to set up the context for the paper. BORO's development and deployment started in the late 1980s. This early work is described in *Business Objects* [3]. BORO's focus was then, and is now, on enterprise modelling; more specifically, it aims to provide the tools to salvage and reuse the semantics from a range of enterprise systems building a single ontology with a common foundation in a consistent and coherent manner.

BORO was originally developed to address a particular need for a solid legacy re-engineering process. This naturally led to the development of methodology for re-engineering existing information systems, currently named bCLEARer – where the capital letters are an acronym for Collect, Load, Evolve, Assimilate and Reuse (we explain these terms later). This aligns with a closely intertwined top-level ontology (the BORO Foundational Ontology). Hence, the term BORO on its own can refer to either of, or both, the mining methodology and the ontology.

Our focus here is on the bCLEARer methodology which is used to systematically unearth reusable and generalized ontological business patterns from existing data. Most of these patterns have been developed for the enterprise context and have been successfully applied in commercial projects within the financial, defense, and energy industries.

bCLEARer has evolved organically over the last three decades both in response to the evolutionary pressures of experience as well as exploiting the opportunities provided by evolving digital technology. An early version of the methodology is described in [3] – with a detailed description in Part 6. At the time this was developed, the late 80s and early 90s, the technology support was immature, so while the process was systematized it was not fully automated. Over the last decade or so, as appropriate technology has emerged, the core process has been fully automated into a data pipeline (described in more detail below). Later versions are described in several places, including [15]. There are also open-source examples on GitHub (https://github.com/boro-alpha).

bCLEARer (and its associated top-level ontology) have, over the years, been configured to exploit a variety of situations ranging from its original legacy system migration to application migration to developing requirements and quality controlling existing systems. A common feature of all these projects has been the initial collection of one or more datasets (where this may include both structured and unstructured data – though structured data is preferred) and its regimented evolution to a more digitally mature state.

IV. Situating digitalization as information evolution

We have established that (engineers have learnt that) the quality of the final product depends upon the engineering quality of the design of the process. We have also suggested that the mainstream ontologization processes are very lightly engineered with a weak background context. This indicates that there is an opportunity to develop a more engineered ontologization

process. What is less clear is what form this engineering should take.

Over a three decades' long evolution, bCLEARer engineering of its practices have been driven by experience. This pragmatic approach is supported by many including Aristotle [16] who said, "for the things we have to learn before we can do them, we learn by doing them". Reflecting upon the process has always been a central part of the practice. However, in the last decade, as the practice has matured, questions about the broader context has naturally arisen and this has led to a much better understanding of how the practice is engineered.

Central to this understanding is the positioning of the practice in the overall evolutionary context. This has, in turn, led to a clearer picture of the specific evolutionary pressures within the practice. From the start, bCLEARer has been framed in terms of information engineering, evolution and revolution. The narrative in [3] was the evolution of information paradigms. For much of its early life the focus of bCLEARer's evolution has been pragmatic practice adapting to the evolutionary pressures presented by actual ontologization with only a modicum of reflection upon the nature of the process. In the last decade or so there has been more reflection on what these practices might reveal. We are reaching a conclusion that the best way to understand the design of the process is make the originally evolutional framing much richer – to firstly more clearly frame the process as digitalization and secondly to show the digitalization as part of a wider trend that sees evolution in terms of information.

In the next section, we set up a broad picture of this wider trend of evolving information. In the section after that we situate digitalization as one transition in that evolution.

1) Broad evolutionary context – situating digitalization as the latest information innovation

Digitalization is an information transition – and it turns out information transition is a universal pattern which can be used to frame the whole of macro-evolution. This provides a reassuringly broad context where digitalization is the latest in a long history of information transitions.

Maynard Smith and Szathmáry [[17] [18] [19]] suggest that macro-evolution can be characterized as a series of information transitions. That one can frame the whole of macro-evolution in these terms:

"… that evolution depends on changes in the information that is passed between generations, and that there have been 'major transitions' in the way that information is stored and transmitted, starting with the origin of the first replicating molecules and ending with the origin of language."

And these changes in information transmission, the passing of "*information … between generations*" are central to evolution. Maynard Smith and Szathmáry provide a table of seven major transitions, where the third is the "genetic code" and the seventh is "language". These not only transform life but also transform the way life evolves – and so, in a sense, evolution evolves through changes in information transmission. They also suggest that each of these transitions has accelerated and expanded evolution enabling more complex entities to emerge quicker.

Jablonka and Lamb, in [20] expanded this framework saying:

"… we argue that information transmitted by non-genetic means has played a key role in the major transitions, and that new and modified ways of transmitting non-DNA information resulted from them."

More specifically, they argue that:

"The evolution of a nervous system not only changed the way that information was transmitted between cells and profoundly altered the nature of the individuals in which it was present, it also led to a new type of heredity—social and cultural heredity—based on the transmission of behaviorally acquired information."

This new type of heredity enables even faster, more flexible, more complex evolution. One key feature is that the social and cultural heredity is not (like genetic heredity) necessarily dependent upon life cycles, so can give rise to adaptations which easily spread through a population within a (genetic) life cycle. This social and cultural adaptation is orders of magnitude faster than genetic evolution and, particularly in changing environments, faster adaptive evolution is more successful. Human culture is a good example of this.

In *Evolution in four dimensions: genetic, epigenetic, behavioral, and symbolic variation in the history of life* [21], Jablonka and Lamb, describe in Chapter 9 - *Lamarckism Evolving: The Evolution of the Educated Guess* how new types of non-genetic heredity – behavioral and symbolic inheritance – enable a new directed evolution, where:

"… the variation on which natural selection acts is not always random in origin or blind to function: new heritable variation can arise in response to the conditions of life. Variation is often targeted, in the sense that it preferentially affects functions or activities that can make organisms better adapted to the environment in which they live. Variation is also constructed, in the sense that, whatever their origin, which variants are inherited and what final form they assume depend on various "filtering" and "editing" processes that occur before and during transmission."

And they label this, understandably, Lamarckian.

a) Further evolutionary transitions in information transmission

This Lamarckian 'targeting' and 'constructing' enables further evolutionary transitions in information transmission. Examples from symbolic evolution are the emergence of writing and printing information technologies. These involve fundamental "changes in the way information is stored and transmitted" where writing involved changes in structure and printing changes in economics. These both clearly led to further innovations in information evolution.

While the transitions clearly involve the use of new technology, closer examination reveals they depended upon the coevolution of human behavior and external information technologies as described by Ong [22] and Olson [23]. Where both the emergence and exploitation of the new technology depends upon intertwined coevolution with human behavior.

There is a similar intertwined evolution of behavior and technology so far in the emergence of computing technology. This current transition is often broadly called, in the context of enterprise processes, 'digitalization' – which includes 'digitization', the process of converting information, data, or physical objects into a digital format, readable by computers.

b) Domestication as example of coevolution

A more familiar example may help us to appreciate the nature of coevolution – the domestication of plants and animals. This has been studied as a distinctive coevolutionary relationship between domesticator and domesticate in a range of research [24], [25]. In this example, it is easy to see that both parties (the domesticators and domesticates) coevolve in the sense of contributing to the relationship. This is perhaps not so easily recognized in the case of the taming of information technology. Zeder [24, p. 3191] describes domestication as a:

"… relationship in which one organism assumes a significant degree of influence over the reproduction and care of another organism in order to secure a more predictable supply of a resource of interest."

Our relationship with the new digital forms of information technology can be described in a similar way. One where we domesticate our computer systems controlling their breeding.

c) A new 'digital' form of information transmission

In *The Selfish Gene*, Dawkins [26] introduced an idea broadly similar to Weismann's lines discussed above. He distinguished between genes as "replicators" that pass on copies of themselves through generations and organisms as "vehicles" or "survival machines" constructed by genes to survive in the environment and so ensure their continued replication.

One could adopt a 'promiscuous ontology', one that regards computer systems as biological individuals [27], [28]] [29]. If so, then computers can easily be seen as "vehicles" for the information they carry and replicate that we domesticate and breed. Furthermore, these biological individuals then pass the Smith and Szathmáry test in so far as they radically 'change the way information is stored and transmitted' directly between themselves – and with humans.

One could be less adventurous and instead see the computer systems as an extension of humans [30]. In this view, computer systems are replicators rather than vehicles – they are part of the apparatus transmitting information rather than "survival machines" in a computer ecosystem. Even on this view, they pass the Smith and Szathmáry test in so far as they radically 'change the way information is stored and transmitted' directly between humans.

So, either way, one can see them as providing an opportunity for an information transition [11].

2) Narrower context – Lamarckian choices for co-evolution

If, as looks likely, history repeats itself and this digital transition follows the pattern of most previous transitions, then it will continue to involve the coevolution of human behavior and digital technology. The energy enterprises currently devote to their digitalization efforts show an intuitive understanding that some kind of directed effort needs to be made. From our evolutionary perspective, we can see this as our culture starting to co-evolve with the new technology – looking to construct the Lamarckian variations that will exploit this opportunity. What is less clear is which Lamarckian constructed variations are likely to lead to significant success – in the language of evolution to be able to exploit the natural selection pressures well.

Maybe history has a clue. A common historical narrative is that technology drives change, that it is the emergence of a new technology that initiates the associated cultural change. Careful study reveals a more interrelated pattern of co-evolution between technology and culture. Where cultural change often prepares the ground for technological innovation and then feeds off it and feeds further innovation. Olson [23] provides a relevant example, explaining how cultural developments in Western Europe from the 9th century onwards played a key role in the invention of moveable type in the 15th century. This technological innovation then laid the ground for developments in Western science in the 16th and 17th centuries. So maybe the coevolution of culture and digital technology evolution can give us some clues on where to target Lamarckian variations.

It is well-accepted that work in logic and mathematics laid the foundation for computing. If we look at the culture of this work, then we can see some trends that help us to target variations to help the co-evolution. Well before the introduction of digital computers, Frege [31] uses the analogy of a microscope and the eye to explain how his formal language compares with ordinary informal language, noting that it provided a superior sharpness of resolution. Carnap [32] talks about 'rational reconstruction' (rationale Nachkonstrucktion). Quine [33] says that one doesn't merely clarify commitments that are already implicit in unregimented language; rather that one often creates new commitments by regimenting. Quine notes that paying attention to the ontological commitment often leads to radical 'foreign' differences [34, pp. 9–10]: "Ontological concern is not a correction of lay thought and practice; it is foreign to the lay culture, though an outgrowth of it" Adding "There is room for choice, and one chooses with a view to simplicity in one's overall system of the world." Lewis [35, pp. 133–5] following the theme of 'outgrowth', argues that the differences are a result of taking the lay common sense seriously, by trying to make it simpler and consistent.

3) The ontologization value chain

The bCLEARer methodology, which is the topic of this paper, has through experience refined these historical intuitions, developing a view of the digitalization process as a network of transformation processes. One way to characterize this is as a value chain, where each transformation adds value. We briefly outline what a value chain is below and then describe the factorization of digitalization into component transformations.

a) Recruiting the value chain view

In manufacturing, Porter's value chain [36] provides a useful tool for broadly characterizing processes as a system of transformations. The system is provided with inputs which feed into a network of transformation processes. This network feeds into the output. The characterization is recursive. Each transformation process can be seen as a sub-system with its own value chain.

We recruit a lightweight version of this tool to characterize ontologization. Under this view, at the broadest level, the ontologization process starts with pre-ontologization information and is transformed using an ontologization process into an ontology. It adds value by transforming the pre-ontologization information into a formal ontology. This reveals an information pathway that starts with the pre-ontologization inputs, undergoes transformations and is output as a formal ontology. Different methodologies have different intermediate transformations and so different information pathways. While the inputs and outputs remain similar, the value-chain transformations differ.

*b) Factoring digitalization into *computerized and *ontologized*

We firstly factorize digitalization into two types of digital transitions that it has found useful to target (and construct). These are *computerization and *ontologization. We use the '*' prefix convention to indicate our specialized use of the term and differentiate it from the many other senses in which it is used.

*Computerization is the process of converting relatively unstructured information into formally structured data. It implies something more than the digitalization mentioned earlier, which just aims at bare computer readability. Formally structured data refers to information that is organized into a highly defined and predictable form, typically within a fixed schema or format. So, a scan of an engineering drawing in, say, PDF format would be digitized but not *computerized, as there is no direct way for the computer to read the components of the drawing. Whereas an engineering drawing in a CAD format, such as native DWG, would be *computerized, as the information in the drawing is explicit in its structure and can be read directly by a computer. *Computerization is intended to be a pragmatic distinction and while there are borderline cases, there are also cases that clearly fall into the pre-*computerization and *computerization camps.

*Ontologization is the process of converting relatively semantically unorganized information into information organized into a common ontological structure. Typically, the information is used in a domain, and there is a level of semantic precision needed for it to be fit for purpose. The *ontologization organizes the information into a common ontological structure that is sufficiently fine-grained to capture the requisite semantic precision.

One way of characterizing *ontologization is that it develops an explicit picture of the ontological commitment [37] [38]. There is a long tradition of seeing this process as a transformation that reveals a deeper structure.

Currently, most information systems being digitized have no precise explicit ontological commitment. So, in practice, the *ontologization is often a regimentation [33] and rational reconstruction [32] of what the ontological commitment would be given some preferred top-level ontology. In bCLEARer's case, the top-level ontology is the BORO Foundational Ontology [4].

*c) *Computerization – transitioning from implicit to explicit formal structure*

The bCLEARer methodology has further identified a factorization of the *computerization transition into two sub-transitions: surface-*computerization and deep-*computerization, corresponding to two levels of *computerization.

The process of transforming unstructured pre-*computerized information, giving it a highly defined and predictable form is sufficient to make it surface-*computerized. Much data in data stores is in this state today. It may, and often in practice does, have significant implicit formal structure. In some cases, making the structure implicit may be deliberate, as part of the process of improving the performance of a system. For our purposes we want to make the structure explicit to facilitate the *ontologization. We do this in the process of deep-*computerization.

Uncovering the underlying implicit form of surface-*computerized information requires a degree of ethnographic hermeneutics – one needs to be able to interpret, to understand, its implicit structure from its perspective. The deep-*computerization transformation aims to expose this and, as far as feasible, make the underlying infrastructure transparently clear.

A simple example of surface-*computerized information would be SQL table schemas and their associated data without the foreign keys noted. This meets the criteria for being *computerized – it has a fixed format. However, when the deep-*computerization adds the foreign keys, one can appreciate that the pre-deep-*computerization information had implicit structure that was not explicitly visible to a computer reading the data. In other words, it was only surface-*computerized.

There are existing software techniques that work in this space, that one can build upon. These include refactoring [39] and clean coding [40]. Both are bodies of practices for restructuring existing code, altering its internal structure to improve it, without changing its external behavior. The restructuring can be recruited to reveal the deeper structure.

From an ethnographical perspective, deep-*computerization (and maybe surface-*computerization too) is a kind of 'surfacing'. As Star notes in *The Ethnography of Infrastructure* [41] the details are technical and "excursions into this aspect of information infrastructure can be stiflingly boring". This means that large parts of the infrastructure are often invisible, in the sense that one doesn't pay attention to them. So, one of the challenges is training oneself to see, and so surface, the invisible structure – a practice with similarities to Bowker's [42] "infra-structural inversion", which foregrounds the backstage operational elements.

As with the other factorizations, this is intended to be a pragmatic distinction where most cases clearly fall into one or other camp – but with some borderline cases. One common borderline case is data cleansing. This includes technical matters such as resolving encoding issues and the treatment of whitespaces (which we find are both still common) as well as keying and spelling errors. While these might degrade the quality of the surface-*computerized information, they do not seem sufficiently grave to undermine its *computerized status. And fixing them does not obviously qualify as immediately revealing implicit structure - though if they are not fixed, they

can hide structure. For pragmatic reasons, we take fixing these to be part of the deep-*computerization process.

d) Inter-process dependency

Obviously, there is an order to the surface- and deep-*computerization process. One surface-*computerizes information before *deep-computerizing it. Theoretically, at least, the *computerization and *ontologization processes would seem to be sufficiently independent that one could undertake either one without the other – implying that there is a choice in which to do before the other.

However, the bCLEARer experience is that there are strong pragmatic reasons for undertaking the full *computerized transition before undertaking the *ontologization transition [38] [11]. Our experience has been that the formalization process inherent in *computerization is best done with raw unaltered data, straight from the operational 'wild'. This is because we found that in cases where the *ontologization process was carried out on pre-*computerization information, it often obfuscated structure that *computerization needed – making the overall process significantly harder. Hence, in bCLEARer we see *ontologization as primarily a process for refining already *computerized data.

V. bCLEARer's broad structure

The bCLEARer process has been modularized (see Appendix B [43], [44]) into a component architectural pattern. We describe this in the first section.

In the earlier comparison of current methodologies, we assessed them relative to two levels: generalization and digitalization. We now describe how bCLEARer addresses these levels in the second and third sections below. In the final, fourth, section we look at whether it is better to surface-*computerize (using bCLEARer) *in vitro or in vivo*.

1) bCLEARer's Pipeline Component Architecture Framework

The bCLEARer process has a pipeline (pipe-and-filter) architecture [44], a prevalent approach for data transformation. This architecture consists of a sequence of processing components, arranged so that the output of each component is the input of the next one creating a 'flow'. The pipeline architecture has, as the 'pipe-and-filter' name suggests, a series of pipe and filter components, where pipes pass data to and from filters that transform the data — the pipeline flow. The architecture can be nested, in that filters can encapsulate a sub-pipeline process.

This generic architectural pattern is refined into a more constrained pattern for bCLEARer's more specific needs. It must include the components of the ontologization process in a structure where the specific arrangement of components can be dictated by the needs of the project and this arrangement can flexibly evolve over time, potentially into a radically different shape.

Typically, it is divided into three broad levels:

1. thin slices – which typically correspond to ways of dividing the domain and the dataset [45]
2. bCLEARer stages – the stages that correspond to a particular type of transformation
3. bUnits level – the filters within a single bCLEARer stage, the base filters are called bUnits.

a) The bCLEARer stage types

While the contents of the individual thin slices and bUnits level vary from project to project depending upon their needs, as well as evolving over time, the bCLEARer stage types are a more stable architectural feature. The design of these types is motivated by the 'separation of concerns' [14] principle – where each type deals with a different kind of transformation. This builds upon the factorization discussed above. The five stage types are Collect, Load, Evolve, Assimilate and Reuse (whose initials contribute to the acronym bCLEARer), which we describe below.

Collect is the stage at which a dataset enters the pipeline. Collect stores the dataset and ensures it is not changed. There is no transformation at this stage. This provides a fixed baseline for tracking. Larger datasets are divided into chunks, to be consumed one chunk at a time.

The Load stage receives the dataset from the Collect stage. The first thing it does is establish the identity of the contents to facilitate tracking and tracing (we explain this later). The Load stage is responsible for ensuring that the data passed onto the next Evolve stage is *computerized – at least surface-*computerized. If the dataset comes from an operational application system, it will probably be sufficiently structured and so need no *computerization transformation. If it is unstructured text, in other words pre-*computerized, then it will need transforming. The Load stage undertakes the minimal amount of transformation to *computerize it, in effect it surface-*computerizes it. Where this is required, the project will need to decide on the output format to use. In our projects, we usually make the target structure simple tables.

The Evolve stage assumes its input data is (at least) surface-*computerized. It is responsible for digitalizing this input data. This is done in two major sub-stages. First it deep-*computerizes the data and then *ontologizes it. Typically, the very first exercise in the deep-*computerize stage is to check whether the data needs cleaning, and if so, clean it. When the data comes from several systems, it normally makes sense as part of the deep-*computerization stage to integrate the data across systems into a common format, as far as possible, after firstly transforming the data from each system on its own. When the deep-*computerization is complete, the *ontologization can start. This is guided by a minimal foundation, the BORO Seed – described later. A full digitalization project will include both *computerization and *ontologization. But pragmatic considerations may dictate that this is done in phases – and the early phases may only go so far along the digitalization journey. For example, undertaking deep-*computerization and delaying *ontologization to a later stage.

The Assimilate stage assumes its input data is 'evolved' – so both locally *computerized and *ontologized. It is responsible for assimilating this into a common cross-project model. The

assimilated model is then ready for use in future assimilate stages.

The Reuse stage assumes its input data is assimilated. It is responsible for translating this data back into a format usable by the targeted operational systems.

b) Managing micro-coevolution

To some extent, the discussions about factorization and components shift focus away from the micro-coevolution that takes place. The bCLEARer journey typically involves evolutionary adaptations simultaneously on two fronts:

- Information Evolution: Adaptation of information throughout its journey.
- Journey Evolution: Adaptation of the journey itself to emerging requirements, accelerating the information's evolution.

The whole process supports both these adaptations: identifying and accommodating significant changes in both the information and its digital journey. A key element is adaptive resilience: maintaining stability and efficiency of the factorization and components amidst continuous change.

2) bCLEARer's level of generality approach

We introduced the broad division of information from a data perspective into levels of generality earlier: metadata, schema and data levels. In our comparison of current methodologies, we mapped their information pathways onto these levels.

a) Accessibility: Shifting level of generality right or left

We can see differences in their approaches from an accessibility perspective. These become visible when one maps the methodologies' information pathways in terms of how they shift accessibility to the three levels of generality left or right along the project plan.

In the 'top-down-classification' approach the metadata level is shifted to the far left, to before the domain ontologization starts. Then it and the schema level are accessible throughout the process. The data level is shifted to the far right, to (presumably) after the project is completed, when the ontology is deployed.

In the 'ask-an-expert' approach the metadata level does not feature, and the schema level is accessible throughout the process. The data level is also shifted to the far right, to after the project is completed.

In the bCLEARer approach accessibility is not restricted at any level. The Collect stage typically starts with a full dataset, one with metadata, schema and data – though the metadata may be implicit.

b) Accessibility: Shifting data right or left

Typically, there is far more data than schema – and more schema than metadata. So, a shift data to the far right will usually have the side effect of significantly reducing the size of the dataset. In the last decades of the 20th century, there may have been good technical performance reasons for working this way, but this is no longer the case.

This approach would also make sense if one developed a high degree of confidence that the schema structures developed would adequately support the requirements of the domain data. If not, then one established this confidence through data testing. And the more one delays the accessibility of the data, the greater the gap between the introduction of a defect and the possibility of finding and fixing it. Hence the development of methodologies such as Extreme Programming [46] that aim to identify defects early. And the emergence of discussion of a shift left approach [47], where testing is performed earlier in the lifecycle. More recently, DevOps has embraced this approach.

c) Top-level ontology deployment

There is an element of ambiguity about the metadata that becomes clear when we consider bCLEARer. In the 'top-down-classification' and 'ask-an-expert' approaches the main information pathway starts with unstructured data, which has no top-level metadata (though it may have provenance 'metadata'). In the 'top-down-classification' approach the top-level metadata is a previously prepared top-level ontology.

When bCLEARer is processing a structured dataset, this will have top-level metadata. During the deep-*computerization stage, this will be made explicit, where it is not already. At this stage one works with the data rather like an ethnographer working within a culture – aiming to make its implicit, invisible assumptions explicit without imposing one's own views, especially on the nature of the domain.

At the *ontologization stage, the situation is different. The ontological commitments are usually far from clear and often the top-level commitments completely inscrutable. bCLEARer gets around this by introducing a top-level, but it does not want to let this interfere with the underlying picture of the domain. So, bCLEARer aims for a balance where the top-level is sufficiently ontologically rich and complex to guide the analysis effectively, but also sufficiently minimal that it does not hinder, or block refinements emerging from the bottom (data) or otherwise render the validation ineffective. One aims to seed the process with a top-level that is sufficient to make the ontological foundations, and so the ontological commitments, scrutable. This could then guide the *ontologization. One also aims to make this as minimal as possible to maximize the benefits of bottom-up grounding. To be as open as possible to refinement as the lower-level ontological commitments emerge from and are confirmed in the data. For an example of a relatively recent minimal seed see *Top-Level Categories* [48].

3) Navigating levels of digitalization – the information pathway

As the two methodologies we looked at earlier show, an ontologization process will have an information pathway that navigates the levels of digitalization. How should one design this navigation?

a) Shifting digitalization right or left

One can characterize this pathway in terms of whether the transformation to *computerization (from unstructured to structured data) is performed earlier or later in the lifecycle (that is moved left or right on the project timeline). The 'ask-an-expert' and 'top-down-classification' approaches, looked at earlier, shift right. They move the transformation to

*computerization towards the end of the project lifecycle (and *ontologize at the same time, without intermediate steps).

bCLEARer shifts left. It aims to make the information pathway transformation to *computerization as early as possible. The ideal case is when the collected dataset is already structured so already *computerized.

What motivates the bCLEARer position is the aim is to be in a situation where the data pipeline can run automatically, getting into 'machines talking to machines' territory as soon as possible. One only needs surface-*computability, not deep-*computability. Hence, one gets all the benefits of automation as early as possible.

b) Example - definitions as a marker of digitalization maturity – shifted right or left

A good marker of the level of digitalization of an approach is how it handles natural language definitions for humans. The level of maturity of this process is a good guide to the overall level of digitalization.

[2] is a good example of a common practice. It has 8 pages (pp. 68-76) and the same number of principles (13-20) devoted to how to write (in natural language) definitions properly. The first principle (13) states: "Provide all nonroot terms with definitions."

From an information pathway perspective, the information is assembled in a human brain and translated into natural language text. Then it is (hopefully) used as an input to a later manual formalization: as the text is not computer readable. Also (hopefully) when the formalization changes, then effort is made to manually bring the natural language text in line. This effort is manual, so is prone to error and does not scale. Clearly, this process is at a pre-*computerization level of digitalization.

Stafford Beer [49] said, "the purpose of a system is what it does" (converted into the acronym POSIWID) and, to rub the point home it has been observed that there is no point in claiming that the purpose of a system is to do what it constantly fails to do. The same point applies, more narrowly, to a computer system's use of a term. If the system (somehow) uses the term in a certain way, then that is surely what the term 'means' to the system. If one writes the code for the system in a 'clean' way [40], we can algorithmically translate the way it uses the term from the machine language of the system into something human readable. Surely this 'definition' with its direct connection with what the system actually does is far more trustworthy than something produced by humans on behalf of the system. And we can also trust that as the system evolves and changes, this 'definition' will change with it.

Hence, bCLEARer aims to clean the data in the pipeline so that the natural language definitions can be algorithmically extracted directly. From an information pathway perspective, all the definitional work is automated, predicated upon at least cleaning the data. This avoids all the manual effort that would have been devoted to these definitions. Clearly, this process is at least at a *computerization level of digitalization, and at an *ontologization level if one needs better output.

*4) Digitalization – surface-*computerization – in vivo versus in vitro*

Directed evolution introduces an *in vitro* mimicry of *in vivo* evolution. This raises a natural question about what factors affect the choice between these two approaches. Obviously, a big factor is how successfully one can target good adaptations.

Given that we have operational enterprise systems, then it is plain that *in vivo* evolution can produce working computer systems (so surface-*computerized data). There is also lots of evidence of failed projects, so we know that it is not easy.

When we have these systems, the problems of dirty data are well-known. If one looks, in any detail, at the data innards of successful enterprise operational systems it is surprising how well they work given how 'dirty' and disorganized they are. Data cleansing exercises show how relatively easy it is to improve them. This suggests that one can target good adaptations for the deep-*computerization process. bCLEARer's experience agrees with this.

When one works with these systems, it soon becomes clear that most have little or no clear ontological commitment. One simple test is to see how the system handles mereology – it is rare to find a system that has a clear picture of this. Similarly for multi-level types [15]. bCLEARer's experience suggests that if done properly then building in a clear ontological commitment is feasible and can reap significant benefits.

Together these suggest that one can target good adaptations for deep-*computerization and *ontologization provided one has the surface-*computerized data. It also suggests that *in vivo* natural selection is not so good at finding these adaptations, at least in the timescale these systems have had to evolve.

From a more general perspective, this suggests that the role of ontologization is not to facilitate the first step in digitalization – surface-*computerization, but more enable a second step that significantly improves systems.

a) Experienced systems

It is normal to think of computer system quality degrading over time, that as systems get older, they accumulate technical debt and the quality of their data declines. It may well be true that as the amount of data in a system grows, the amount of erroneous data items also grows. But this misses an important point from our perspective, that systems accumulate a kind of experience over time. Typically, both the variety and complexity of their data increases and it becomes a better reflection of the domain. This reflects the enormous investment in both the operation and maintenance of the system.

From the bCLEARer perspective this 'experience' is valuable. As Ashby's discussion [50] of variety makes clear the richness and complexity of the picture we build of the domain will depend upon the richness and complexity (the variety) of the data we use to build it.

From one perspective, this is a classical evolutionary situation. A biological unit (in this case, a computer system) garners information about its world that is useful to it. Unless this information is heritable, and inherited, then it stops being useful when the unit dies. Genetic inheritance is a very lossy way of transmitting information. Pipelines like bCLEARer offer the prospect of salvaging significant portions of the data.

From another, breeding perspective, it suggests a rule of thumb. Given that we aim to harvest domain patterns from the digitalization exercise, then if we select more experienced operational systems – and several of them – then we will harvest richer more accurate patterns.

*b) Harvesting pre-*computerization*

There is a flip side to this. What should we do in cases where there is not even an operational system, let alone an experienced one. We can deploy ontologization processes on pre-existing unstructured data or even synthesize unstructured data. The synthesized data will not have been subjected any real selection pressures. The unstructured data may have been subject to some selection pressures, but these will not shape its formal structure (as it is unstructured). In these cases, there seems to be a lack of variety, or at least the right kind of variety.

A rationalist might think armchair reflection will be able to provide requisite variety. But this will be rooted in brain wetware, speech and writing – all legacy technologies from a computing perspective. Is this good enough to target good, computerized adaptations? There is a lack of data on this topic, but our anecdotal evidence is that it falls well short of what is required. Without the computer experience to build upon, our targeting falls back on legacy technology patterns of thought that prove unsuitable for computerization.

VI. SITUATING BCLEARER AS EVOLUTION

One can see the goal of a bCLEARer project is to build an *in vitro* high evolvability environment for the information, giving it the possibility of evolving fast. This environment is easier to build if one has a sensitivity to the drivers that the evolutionary perspective reveals – and for this one needs to adopt the perspective.

The evolutionary perspective is an incredibly rich resource, and we are still in the process of understanding how the digitalization process fits into it. However, there are several elements of the perspective that we have found useful and we outline a few of these in this section to provide a sense of what the perspective entails.

1) bCLEARer as directed or experimental evolution

From an evolutionary perspective, the bCLEARer methodology can be seen as a kind of directed [51] (or experimental [52]) evolution – where experimental evolution is sometimes called "laboratory natural selection".

Direct evolution is used in protein engineering – and contrasted with rational design, which targets specific point mutations. However, from a broader perspective, the range goes from random natural mutation to deterministic rational design with directed evolution somewhere in the middle. But the rate of natural mutation is usually insufficient for generating the genetic diversity required for laboratory directed evolution, so it is out of scope. Directed evolution balances the difficulty of 'rationally' accurately predicting how specific mutations will impact protein function with the slow and unpredictable pace of natural selection. It uses targeting but reduces the need for accurate predictions replacing by iteratively selecting mutations and 'empirically' testing them.

In the digitalization context, rational design would be attempting to build the ontologization from first principles, whereas the directed evolution would start with the existing data and target a range of likely mutations and iterative tests which lead to better adaptations. In a process broadly analogous to directed and experimental evolution, it repeatedly targets and constructs variants in an iterative step-by-step process, continually inspecting the results and selecting for fitness, aiming to mimic natural selection.

This evolution is directed in the sense that there is an element of Lamarckian target setting when managing the mutations, which when successful speeds up the evolution. The direction cultivates and nurtures evolution—focusing on fostering and guiding innovation —rather than purely analyzing or breaking down data. The aim is to guide the selected dataset (of information) along a journey of digitalization transformation that exploits the opportunities offered by digital technology. More specifically, to exploit the opportunities for *computerization and *ontologization of information.

2) bCLEARer as evolving information transmission

Inheritance involves the transmission of information between individuals. Genetic heredity involves passing information from one generation to the next – often called vertical transmission. Sexual reproduction is an example of vertical transmission – transmission from parents to their offspring.

If, as suggested earlier, we see computer systems as biological individuals, then within digitalization, legacy system replacement can be seen as a good example of vertical transmission, where the data (information) in the legacy system is transmitted to the new system.

The bCLEARer pipeline works at the level of information transmission – in other words, transmission between systems. If deployed in the legacy system replacement case, it would take control of the transmission of information from the legacy to the new system. So, the bCLEARer pipeline can be seen as an example of digitalization information transmission, with a focus on developing adaptations during transmission between computers.

Developing adaptations during information transmission is not new to evolution. In natural evolution mechanisms for creating adaptations during information transmission have arisen, sexual reproduction being a classic example. The genetic recombination of genetic material from two parents can introduce novel variation.

bCLEARer pipelines are designed for fast evolution. For good empirical reasons, biological experimental evolution, mentioned above, will often adopt a life span speed strategy. It will select individuals, such as fruit flies, with a short life span, to enable testing to occur over multiple generations and so speed up evolution. However, this strategy of shortening life spans and increasing the number of generations makes less sense in the bCLEARer case, where (among other things) there is not a plethora of systems with short lifespans. However, the general strategy of increasing the pace of evolution stands. bCLEARer achieves this through both extending and enriching the information transmission process as well as iterating it –

mimicking the evolution of multiple generations within a single transmission.

3) Challenge – uncertainty, contingency and chance

The uncertainties around innovation are well-known [53]. One way these uncertainties are framed in evolution is as contingency [54] [55], This recognizes that evolution is a historical process and so is sensitive to and so contingent upon, the paths taken in its journey – in other words, sensitive to chance.

In general, the potential for innovation is usually so wide-ranging that it makes no real sense to ask whether an opportunity has been missed. However, in the restricted digitalization context where we are establishing patterns of adaptations that lead to systematic innovations, it makes more sense to ask whether we are missing innovation opportunities that we could (should?) have spotted. In this context, framing the uncertainty as contingency usefully highlights the question of whether we successfully exploit an innovation opportunity.

a) Macro- and micro-evolutionary contingency

To illustrate evolutionary contingency, Stephen Jay Gould [56] used the thought experiment of rewinding the "tape of life" to the distant past. He argued that even small changes to the path of history could result in evolutionary outcomes very different from our world, such as, for example, no humanity.

For our purposes, we can usefully distinguish between broadly global macro-contingency and local micro-contingency. Where global macro-contingency is whether a particular major outcome will ever (globally) happen – for example, humanity or language or computers emerging. And local micro-contingency is whether a particular minor outcome that could happen will do so in a local situation. We see micro-evolutionary contingency when different similar beetle populations respond differently to the same pressures – such as climate change.

b) Macro- and micro-evolutionary bCLEARer contingency

We can translate macro- and micro-evolutionary challenges to our bCLEARer digitalization context.

At the macro-evolutionary level, we recognize that it is not inevitable that humanity will exploit the major opportunities of digital technology. We have already noted that the exploitation of technology depends upon the appropriate co-evolution of technology and cultural practices. And that the evolution of the cultural practices depends, at least in part, upon the appropriate Lamarckian targeting. If this doesn't happen, the innovation opportunity will be missed. The (broadly) global question asks whether it will happen in our (near) future.

The concern is not entirely theoretical as we have examples from the past. Olson [23] also describes how Western European culture successfully evolved to take advantage of printing technology when cultures in other parts of the globe (such as China) did not, even though they had earlier access to the technology. This provides us with a good illustration that technological innovations need cultural variations that will successfully exploit selection pressures, that these evolutionary pathways are contingent upon taking (targeting and construction) a potentially successful direction.

Pragmatically, contingency concerns are about completeness – about how exhaustive the process is. At the macro level, the goal is to design a framework whose use is likely to maximize the chances of finding and exploiting the general opportunities, particularly the most fruitful opportunities, for *computerized and *ontologized digitalization. At the micro level, the goal is to design a pipeline using the framework whose operation is likely to maximize the chances of finding and exploiting the specific opportunities. At both levels, the aim is to minimize the risk of overlooking valuable opportunities.

bCLEARer is an example of such a framework at both the macro- and micro-levels. It is designed as a tool to systematically find and exploit opportunities for *computerized and *ontologized digitalization.

4) bCLEARer as in vitro evolution

If one restricts one's perspective to the bCLEARer pipeline, then most of the process is (in a sense) *in vitro* – in a walled garden outside the original system. However, if one steps back the bCLEARer pipeline usually plays a role in a wider live *in vivo* ecosystem.

Also, there are usually important (in a sense) *in vivo* tests where the information is returned to an operational system and tested 'in the wild'. If possible, this is both the original and similar systems. In an ideal configuration of bCLEARer, the improvements are fed back into the original system on an ongoing basis and the results inspected.

VII. Outlier Design Choices

In the bCLEARer information transmission there are two evolutionary processes each with their own information pathway. There is the information being processed by bCLEARer and then there in the bCLEARer process itself – as code. Both are co-evolving intertwined in a process of reciprocal causation – each feeding of the other.

The information being processed by bCLEARer is the full data set, with no level of generality being excluded. The bCLEARer process is as automated as possible, so its information pathway as *computerized as possible – in other words, is digitalization shifted left as far as possible. Both these are outlier design choices. In this section we look ways of explaining aspects of these outlier choices.

1) Should transmission include data inheritance

There is a further refinement of the evolving information transmission narrative related to the role data plays in it. In the methodologies we have been looking at, under the level of generality accessibility perspective, one can choose whether to include data (the lowest level of generality) in the ontologization process. Simplifying a little, the 'ask-an-expert' and 'top-down-classification' methodologies exclude data, the bCLEARer methodology includes it from the start. (The simplification is that the metadata-data-schema classification is about the syntax of the implementation, whereas generality is a semantic matter. However, there is good enough rough match between syntax and semantics here to make the point fair.) This is an all or nothing choice. Unlike in software development methodologies, especially waterfall, where the full dataset is included in the process part of the way through – typically towards the end.

Hence, we label this as an architectural design choice on whether to shift (far) left or (far) right.

a) Weismann's distinction

One can get a sense of this architectural design question from a distinction make in 19th century evolution theory. Weismann [57] turned the point that the mechanisms of transmission typically can only transmit some information in the source into a distinction. He made a basic (since refined) division of cells into the germline and the somatic line which gives us a neat, simplified picture of the underlying structure.

The germline is those cells that are involved in reproduction and the transmission of genetic information from one generation to the next. Mutations in the germline are crucial for evolution because they can be passed to the next generation, in other words, they are heritable.

The somatic line is the rest of the cells, the non-reproductive cells. They are not involved in the same way in transmission. Mutations in these cells may affect the individual and so their fitness, but are not transmitted on to offspring, so they are non-hereditary.

The germline cells have been called 'immortal' in the sense that they (or their genes) continue to exist indefinitely through reproduction – creating a lineage. Whereas the individuals and their somatic line cells die, they are mortal. Thus, changes in the germline can contribute to genetic diversity and evolutionary adaptation, while changes in somatic cells affect only the individual organism's health or fitness.

If one maps Weismann into the world of computer systems, then there is a recurring pattern of transmissions where the data-schema division aligns with the germ-somatic line division, where data is heritable, and schema is not.

One clear example is the migration between COTS systems which has an analogous structure to vertical genetic transmission. The data is migrated (transmitted) from the old application to the new application – and so is immortal in the sense it persists between generations. The schema (and the rest of the application) is like the somatic line in that it 'dies' with the old application. APIs (Application Programming Interfaces) also have an analogous transmission structure. The data is transmitted between applications whereas the schema is not.

This data-immortal, schema-mortal picture is, like Weismann's, a simplification. But it is broadly true in that the data persists much longer than the schema – though as it moves between applications it gets mapped to the new applications schema.

We can frame this in economic terms. If we think of a biological unit (whether organic or silicon computer application) as storing information as an investment, one which 'pays' a return when used. Then information transmission can be seen as a way of preserving that investment across units to generate better 'returns. When this insight is combined with the realization that in many current transmissions data is transmitted and schema is not, then data would seem to be a better place to invest in the information system ecosystem.

2) Data as embodied competencies

The use of competency questions is a rationalist approach. At its simplest, it assumes that we have sufficient knowledge to unaided target competencies that we require and then construct a computer system with them competencies.

bCLEARer is an example of an empirical approach. It starts with source operational systems that we can verify have a certain level of competence. The datasets from these systems embody these competences. They must do, otherwise the systems would not operate. One can make these competencies explicit, exhibit them, through queries on the datasets – ones which are often already built into the source systems.

Over time, the data structures in computer systems are twisted and turned to accommodate new requirements. Hence, there is an understandable feeling that the structures are somehow defiled, unclean. While it is probably true that the lack of cleanliness holds back some level of competency, it is not true that it indicates a (total) lack of competency. The computer systems operate, often at a sophisticated level, they still have the competencies. The *ontologization stage of the digitalization process addresses this lack of cleanliness providing a hyper hygienic level of cleanliness that lets new competencies emerge.

3) Managing the inheritance - preserving and improving the investment

If one shifts right and includes data in the process, then one is faced with a responsibility for managing that data.

a) Transmission fidelity and transformations

Transmission fidelity ensures that the transmitted information maintains its original shape and characteristics throughout the transmission. In genetic (DNA) inheritance a reasonably high transmission fidelity is needed to maintain organismal stability across the generations. But mutations, a failure of fidelity, are the variations that provide the raw material for evolution. So, if we want adaptation and natural selection to occur we need to ensure we have mutation and so variation.

In the pipeline, the formal nature of digital computing means fidelity works in a different way. Though there is some degradation of the digital signal in some circumstances, this is not significant. So, we can pragmatically assume digital fidelity. We still need new variations, but these are formally created by the pipeline code.

DevOps recognizes the importance of pipeline observability engineering [58]. The term is borrowed from control theory, where the "observability" of a system measures how well its state can be determined from its outputs. Majors et al. [58] suggests that what differentiates observability is its focus on not just identifying issues but aiming to minimize the amount of prior knowledge needed to resolve an issue. This has, historically, been a significant driver for bCLEARer where significant time used to be lost attempting to track and trace adaptations along the information pathway. Where tracking follows information and tracing follows how information has influenced other information. This has led to the bCLEARer pipeline introducing an additional kind of observability – what we call 'inspectability' – which is the ability to map in full detail the transformation journey along the information pathway.

We have been working over the last few decades evolving an inspectability framework. The specific goal of this inspectability framework is to be able to track and trace the items of information through the pipeline. This relies firstly on having a clear notion identity for these items. This means, ironically, we needed to build an ontology for the information in the pipeline. We need to be able to extend this ontology to give us a clear notion of tracing – relating how items are transformed into new items. We then needed to build infrastructure into the pipeline to make this ontology explicit. Finally, we needed to be able to access this ontology at regular inspection gates and have tools that allow us to view and visualize it.

With this in place we can track and trace information items and their transformations through the pipeline, between pipeline runs and between pipeline evolutions. One useful visualization is the information items' ontogenic tree – analogous to the phylogenetic tree – showing how the information items transform as they pass along the information pathway, as well as how data and schema coevolve.

b) Automation and the dataset

Automation has improved the pace and scale of digitalization's directed evolution. It is well-known that pace is a key factor in being able to generate change in a reasonable time. In evolutionary research the fruit fly has a key role due to a very short life cycle, typically around 10 days from egg to adult, leading to fast evolution. In innovation research, Christensen (see *The Innovator's Dilemma* [53]) picked the disk drive industry because of its fast pace of change, referring to it as the 'fruit fly' of the business world.

Pace is similarly important in ontologization pipelines. It has a couple of aspects which we have already noted a few times. The first and simplest is the pace of a single pipeline run - the information evolution. This needs to be quick enough to allow for frequent runs. The second is the pace of the evolution of the transformations in the run in a project – the project process evolution. The third is the pace of the evolution of the transformations across projects – the process evolution.

A major impact on pace, as well as the investment required, is the development of the pipeline code. An important way to reduce cost was to evolve common code, where code is reused rather than written anew from scratch. The aim is for much of the final code to be common to multiple bCLEARer projects. There are opportunities to build common code for the running of the pipeline. There are also opportunities to evolve general patterns of transformation (and the components of the transformations). One can see this as digitalizing the transformations – where the transformations are carried out by machines on machines. Using machines to build better machines has a long history. One well-known episode is the use of John "Iron-Mad" Wilkinson's machine to precisely bore the large cylinders needed for James Watt's steam engines – significantly improving efficiency over the previous manually crafted ones.

Achieving the goal of a common codebase requires the adoption and coordination of multiple techniques. There are a variety of software development hygienes that reduce the cost of maintenance and enhancement such as clean coding [40]. There is also the continuing evolution of design patterns facilitated by a close analysis of the transformations.

c) Example design pattern – unification of types

One useful design pattern simplifies the handling of data formats. There is no restriction of the format in which the dataset comes in at the Collect stage. It could be in XML, JSON or SQL or a combination of these or other formats.

When this reaches the Evolve stage, it will be deep-*computerized. This phase will be approached with an ethnographic mindset, interpreting and understanding its implicit structure from its own perspective – aiming not to introduce any biases. For the ontologization process the implementation in the data format is an irrelevancy – only the structure is relevant. This creates an opportunity for a variation that has no impact. Higher levels of (syntactic) generality, the metadata and schema can be mapped into the data, which removes the dependency on any specific implemented format. We call this mapping the unification of types [27], [59]. This enables us to choose for this stretch of the pipeline the implemented data format that suits the work we want to do – and build common code for this. It also greatly simplifies making the metadata explicit – as what was built implicitly into the implemented data form can now be made explicit.

VIII. Conclusion

The paper has, using bCLEARer as an example, indicated several ways of extending the design space of ontologization practices. It has , including the shift far left of *computerization and the accessibility of data. It has also introduced contextual scaffolding to provide an evolutionary perspective that situates digitalization as the latest iteration in the overall evolution of information transmission. And then situates *computerization and *ontologization as key cultural practices in digitalization's coevolution.

Acknowledgment

We wish to thank Andreas Cola for his helpful review of the paper.

References

[1] N. Guarino and C. Welty, 'Identity, unity, and individuality: Towards a formal toolkit for ontological analysis', in *Proceedings of the 14th European Conference on Artificial Intelligence*, Citeseer, 2000, pp. 219–223.

[2] R. Arp, B. Smith, and A. D. Spear, *Building ontologies with Basic Formal Ontology*. Cambridge, Massachusetts: Massachusetts Institute of Technology, 2015.

[3] C. Partridge, *Business Objects: Re-Engineering for Re-Use*, 1st Edition. Oxford: Butterworth-Heinemann, 1996.

[4] S. de Cesare and C. Partridge, 'BORO as a Foundation to Enterprise Ontology', *Journal of Information Systems*, vol. 30, no. 2 (Summer 2016), Art. no. 2 (Summer 2016), 2016, doi: 10.2308/isys-51428.

[5] J. Mokyr, ‘The past and the future of innovation: Some lessons from economic history’, *Explorations in Economic History*, vol. 69, pp. 13–26, Jul. 2018, doi: 10.1016/j.eeh.2018.03.003.
[6] J. Mokyr, *A culture of growth: the origins of the modern economy*. in The Graz Schumpeter lectures. Princeton, NJ: Princeton University Press, 2017.
[7] W. E. Deming, *The new economics: for industry, government, education*, 2. ed. Cambridge, Mass.: MIT Press, 2000.
[8] J. M. Juran, *Quality Control Handbook*. New York: McGraw-Hill, 1951.
[9] C. Dutilh Novaes, ‘The Formal and the Formalized: The Cases of Syllogistic and Supposition Theory’, *Kriterion*, vol. 56, no. 131, Art. no. 131, Jun. 2015, doi: 10.1590/0100-512X2015n13114cdn.
[10] C. Partridge, ‘Digitalisation Levels’, Gemini Call, Weekly meeting, 2021. [Online]. Available: https://www.academia.edu/89132134
[11] C. Partridge, ‘How an Evolutionary Framework Can Help Us To Understand What A Domain Ontology Is (Or Should Be) And How To Build One’, presented at the FOMI 2022, 12th International Workshop on Formal Ontologies Meet Industry, 12-15 September 2022, Tarbes, France, Sep. 12, 2022. [Online]. Available: https://www.academia.edu/94453259
[12] A. Fernández-Izquierdo, ‘Methodological framework for ontology management’, OntoCommons, D4.2, 2021. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.10890089
[13] M. Gruninger and M. S. Fox, ‘Methodology for the Design and Evaluation of Ontologies’, in *International Joint Conference on Artificial Intelligence*, 1995. [Online]. Available: https://api.semanticscholar.org/CorpusID:16641142
[14] E. W. Dijkstra, ‘On the role of scientific thought’, in Selected Writings on Computing: A personal Perspective. , New York, NY: Springer New York, 1982, pp. 60-66–362. doi: 10.1007/978-1-4612-5695-3_12.
[15] C. Partridge *et al.*, ‘Implicit requirements for ontological multi-level types in the UNICLASS classification’, in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2020, pp. 1–8.
[16] Aristotle, *Nicomachean Ethics*, 3rd ed. Hackett Publishing, 2019.
[17] J. Maynard Smith and E. Szathmáry, *The origins of life: from the birth of life to the origin of language*, Reprinted. Oxford: Oxford Univ. Press, 2009.
[18] E. Szathmáry and J. M. Smith, ‘The major evolutionary transitions’, *Nature*, vol. 374, no. 6519, Art. no. 6519, Mar. 1995, doi: 10.1038/374227a0.
[19] J. Maynard Smith and E. Szathmáry, *The major transitions in evolution*. Oxford: Oxford University Press, 2010.
[20] E. Jablonka and M. J. Lamb, ‘The evolution of information in the major transitions’, *Journal of Theoretical Biology*, vol. 239, no. 2, Art. no. 2, Mar. 2006, doi: 10.1016/j.jtbi.2005.08.038.
[21] E. Jablonka and M. J. Lamb, *Evolution in four dimensions: genetic, epigenetic, behavioral, and symbolic variation in the history of life*. in Life and mind. Cambridge, Mass: MIT Press, 2005.
[22] W. J. Ong, *Orality and literacy: the technologizing of the word*. 1982.
[23] D. R. Olson, *The World on Paper*. 1994.
[24] M. A. Zeder, ‘Core questions in domestication research’, *Proc. Natl. Acad. Sci. U.S.A.*, vol. 112, no. 11, Art. no. 11, Mar. 2015, doi: 10.1073/pnas.1501711112.
[25] G. Larson and D. Q. Fuller, ‘The Evolution of Animal Domestication’, *Annu. Rev. Ecol. Evol. Syst.*, vol. 45, no. 1, Art. no. 1, Nov. 2014, doi: 10.1146/annurev-ecolsys-110512-135813.
[26] R. Dawkins, *The selfish gene*. New York: Oxford University Press, 1976.
[27] C. Partridge, ‘Why Form, and so Unification of Types, is Important’, presented at the King’s College London, Workshop on the Unification of Types and Multi-Level Modeling, 13 March 2024, London, UK, Mar. 13, 2024. [Online]. Available: https://www.academia.edu/116276110
[28] J. Dupré, *Processes of Life: Essays in the Philosophy of Biology*. New York: Oxford University Press UK, 2011.
[29] T. Pradeu, ‘The many faces of biological individuality’, *Biol Philos*, vol. 31, no. 6, Art. no. 6, Nov. 2016, doi: 10.1007/s10539-016-9553-z.
[30] A. Clark and D. J. Chalmers, ‘The Extended Mind’, *Analysis*, vol. 58, no. 1, Art. no. 1, 1998, doi: 10.1093/analys/58.1.7.
[31] J. Van Heijenoort, Ed., ‘Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought: GOTTLOB FREGE(1879)’, in *Frege and Gödel*, Harvard University Press, 1970, pp. 1–82. doi: 10.4159/harvard.9780674864603.c2.
[32] R. Carnap, *The Logical Structure of the World*. Chicago and La Salle, Ill.: Open Court, 1967.
[33] W. V. O. Quine, *Word & Object*. MIT Press, 1960.
[34] W. V. Quine, Ed., *Theories and Things*. Cambridge: Harvard University Press, 1981.
[35] D. K. Lewis, *On the plurality of worlds*. Malden, Mass: Blackwell Publishers, 1986.
[36] M. E. Porter, *Competitive advantage: creating and sustaining superior performance*. New York: Free Press : Collier Macmillan, 1985.
[37] P. Bricker, ‘Ontological Commitment’, in *The Stanford Encyclopedia of Philosophy*, Winter 2016., E. N. Zalta, Ed., Metaphysics Research Lab, Stanford University, 2016. [Online]. Available: https://plato.stanford.edu/archives/win2016/entries/ontological-commitment/
[38] C. Partridge, ‘Why (and how) to use a metaphysicalist foundational ontology’, 2015.

[39] M. Fowler and K. Beck, *Refactoring: improving the design of existing code*, 28. printing. in The Addison-Wesley object technology series. Boston: Addison-Wesley, 2013.
[40] R. C. Martin, *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009.
[41] S. L. Star, 'The Ethnography of Infrastructure', *American Behavioral Scientist*, vol. 43, no. 3, Art. no. 3, Nov. 1999, doi: 10.1177/00027649921955326.
[42] G. C. Bowker, *Science on the run: information management and industrial geophysics at Schlumberger, 1920-1940*. in Inside technology. Cambridge, Mass: MIT Press, 1994.
[43] C. Partridge, A. Mitchell, and P. Grenon, 'A Framework for Composition: A Step Towards a Foundation for Assembly', CDBB, Apr. 2021. doi: 10.17863/CAM.66459.
[44] D. Garlan and M. Shaw, 'An Introduction to Software Architecture', in *Series on Software Engineering and Knowledge Engineering*, vol. 2, WORLD SCIENTIFIC, 1993, pp. 1–39. doi: 10.1142/9789812798039_0001.
[45] C. Partridge, 'Developing Thin Slices: An Introduction to the Methodology for Developing the Foundation Data Model and Reference Data Library of the Information Management Framework', (draft), Mar. 2022.
[46] K. Beck, *Extreme Programming explained: embrace Change*, 9. print. in The XP Series. Boston, Mass. Munich: Addison Wesley, 2003.
[47] L. Smith, 'Shift-left testing', *Dr. Dobb's Journal*, vol. 26, no. 9, Art. no. 9, 2001.
[48] C. Partridge, 'Top-Level Categories: Categories for the Top-Level Ontology of the Information Management Framework', 2022.
[49] S. Beer, 'What is cybernetics?', *Kybernetes*, vol. 31, no. 2, Art. no. 2, Mar. 2002, doi: 10.1108/03684920210417283.
[50] W. R. Ashby, *An introduction to cybernetics*, 6. repr. London: Chapman & Hall, 1979.
[51] L. Sellés Vidal, M. Isalan, J. T. Heap, and R. Ledesma-Amaro, 'A primer to directed evolution: current methodologies and future directions', *RSC Chem. Biol.*, vol. 4, no. 4, Art. no. 4, 2023, doi: 10.1039/D2CB00231K.
[52] T. J. Kawecki, R. E. Lenski, D. Ebert, B. Hollis, I. Olivieri, and M. C. Whitlock, 'Experimental evolution', *Trends in Ecology & Evolution*, vol. 27, no. 10, Art. no. 10, Oct. 2012, doi: 10.1016/j.tree.2012.06.001.
[53] C. M. Christensen, *The innovator's dilemma: when new technologies cause great firms to fail*. in The management of innovation and change series. Boston, Mass: Harvard Business School Press, 1997.
[54] J. Beatty, 'The evolutionary contingency thesis', in *Concepts, Theories, and Rationality in the Biological Sciences: The Second Pittsburgh-Konstanz Colloquium in the Philosophy of Science, University of Pittsburgh, October 1-4, 1993*, UVK, Universitätsverlag Konstanz, 1995, p. 45.
[55] J. Beatty, 'Chance and Natural Selection', *Philosophy of Science*, vol. 51, no. 2, Art. no. 2, 1984.
[56] S. J. Gould, *Wonderful life: the Burgess shale and the nature of history*. New York London: W. W. Norton, 1989.
[57] A. Weismann, *Essays upon heredity and kindred biological problems*. Oxford: Clarendon Press, 1889.
[58] C. Majors, L. Fong-Jones, and G. Miranda, *Observability engineering: achieving production excellence*, First edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2022.
[59] C. Partridge, 'Unification of Types and Multi-Level Modeling: Introduction – IS', presented at the King's College London, Workshop on the Unification of Types and Multi-Level Modeling, 13 March 2024, London, UK, Mar. 13, 2024. [Online]. Available: https://www.academia.edu/116275287

(KBSI. 1994) KBSI. 1994. IDEF5 Method Report.

(Uschold. 1995) Uschold, Mike, and Martin King. 1995. 'Towards a Methodology for Building Ontologies'.

(Jones. 1998.) Jones, Dean, Trevor Bench-Capon, and Pepijn Visser. 1998. 'Methodologies for Ontology Development'.

(Fernández-López. 2002) Fernández-López, Mariano, and Asunción Gómez-Pérez. 2002. 'Overview and Analysis of Methodologies for Building Ontologies'. The Knowledge Engineering Review 17(2):129–56. doi: 10.1017/s0269888902000462.

(Cristani. 2005) Cristani, Matteo, and Roberta Cuel. 2005. 'A Survey on Ontology Creation Methodologies'. International Journal on Semantic Web and Information Systems 1(2):49–69. doi: 10.4018/jswis.2005040103.

(Iqbal. 2013) Iqbal, Rizwan, Masrah Azrifah Azmi Murad, Aida Mustapha, and Nurfadhlina Mohd Sharef. 2013. 'An Analysis of Ontology Engineering Methodologies: A Literature Review'. *Research Journal of Applied Sciences, Engineering and Technology* 6(16):2993–3000. doi: 10.19026/rjaset.6.3684.

Commented [CP2]: To be updated