MIS Quarterly

RESEARCH ARTICLE

FROM BITS TO ATOMS: OPEN SOURCE HARDWARE AT CERN[1]

Laia Pujol Priego
Ramon Llull University, ESADE Business School,
Barcelona, SPAIN {Laia.pujol@esade.edu}

Jonathan Wareham
Ramon Llull University, ESADE Business School,
Barcelona, SPAIN {Jonathan.wareham@esade.edu}

Although considered a relatively recent phenomenon of the past decade, open source hardware (OSH) is already influencing commercial hardware development. However, a common belief is that the greater economic cost and complexity of hybrid digital objects (i.e., digital objects with both hardware and software) precludes their development with open source methods traditionally used for software. We study a sophisticated OSH named White Rabbit initiated at CERN and developed through a vibrant and heterogenous open source community. Our findings show that the assumption that hardware and software require fundamentally distinctive development and production modes should be replaced with a more nuanced differentiation characterized by three main attributes describing an object's composition: embodiment, modularity, and granularity. Taken together, these three attributes determine how a hybrid object is developed throughout its evolution in an open source community. Our research offers several contributions. First, we provide a more nuanced view of the consequences of the material embodiment of hardware. Once considered a simple deterrent to open source development, we describe how economic cost is subordinate to more influential aspects of an object's physical layers: as the open source community modifies the object to accommodate the operating requirements of diverse physical instantiations, such modifications can be incorporated in the logical design covered by the open source license. Additionally, we show how embodiment, modularity, and granularity progress through the object's evolution and how this maturation subsequently affects development modes. We trace the implications of our findings for hybrids and digital object conceptualizations in IS research, open source development and, more broadly, normative implications for OSH in scientific and commercial computing.

Keywords: Open source, open source hardware, hybrid objects, digital objects

Introduction

Open source hardware (OSH) is a term for tangible artifacts—machines, devices, or other physical things—for which the design is made publicly available in a way that enables anyone to study, modify, distribute, make, and sell either a design or hardware based on the design (Open Source Hardware Association, 2012). The rise of the do-it-yourself (DIY) phenomenon with physical maker spaces, hackspaces, and FabLabs (Gibb, 2014), along with the general proliferation of OSH, has attracted the attention of numerous organizations that include scientific research infrastructures needing customizable scientific hardware, industrial organizations, and traditional hardware manufacturers (Balka, 2011; Boisseau et al., 2018; Mellis & Buechley, 2012; Pearce, 2012). Although considered a relatively recent phenomenon of the past decade (Balka et al.,

[1] Sirkka Jarvenpaa was the accepting senior editor for this paper. Jan Recker served as the associate editor.

2010; Bonvoisin et al., 2017), OSH[2] is already influencing commercial hardware development. Some compelling examples include Arduino, RepRap, the Open Compute Project, and RISC-V.[3]

Notwithstanding the growing interest in OSH, this phenomenon poses the question of whether the premises of *open source development*, which have historically been built around the development of software (Crowston & Howison, 2006; Feller & Fitzgerald, 2001; Fitzgerald, 2006; Mockus et al., 2002), apply to a form of technology *object*[4] (i.e., a *hybrid*) which is both (1) a *material object* with a "physical mode of being" (Faulkner & Runde, 2013, p. 806) and (2) a *digital* object containing syntactic entities with one or more bitstrings (Faulkner & Runde, 2019). *Hybrids* (Faulkner & Runde, 2009, 2013, 2019) include any types of hardware (i.e., electro-mechanical devices) that contain middleware or embedded software (Yoo, 2010).

Early research in OSH has explored the specific challenges of transposing open source development—characterized as highly voluntary, with loosely centralized, parallel collaborations—to the development of hybrids (Dahlander & O'Mahony, 2010; Feller & Fitzgerald, 2000, 2001; Fitzgerald, 2006; Howison & Crowston, 2014; Lindberg et al., 2016; Shah, 2005, 2006). Challenges include long developmental cycles and slower iterations of patches and improvements to physical prototypes (Boisseau et al., 2018), excessive complexity (Oberloier & Pearce, 2018), limitations in the available software for hybrid digital development, financial costs of development and production, and insufficient licensing of OSH (Balka et al., 2010; Balka, 2011). Fundamentally, these studies assume that the physical nature of hybrids is the main constraint in transposing open source development to hybrids. In parallel, there are growing calls to develop a more subtle understanding of the nature of hybrids: "for information systems theory and practice, the confluence of the digital and physical is a largely unexplored territory worth exploring, as it has the potential to fundamentally change our environment" (Kyriakou et al., 2017, p. 327). However, excessive attention to the physical nature of OSH may overshadow a more useful and comprehensive explanation of how a hybrid object's attributes affect the developmental model. Given the expectations that OSH "bears an enormous potential for reframing the social organization of product development and therewith to disrupt conventional industrial practices" (Mies et al., 2019, p. 129), we formulate the main research question of our paper:

How do the attributes of a hybrid object and its components affect the open source model of development?

The objectives of this paper are: (1) to provide an empirical description of the process of developing a hybrid object through open source development, and (2) to integrate the empirical findings into a theoretical model that explains how the attributes of hybrids and their components affect open source development. To achieve these objectives, we conducted a qualitative case study of a high-profile OSH. White Rabbit (WR) is a hybrid object developed by CERN (Conseil Européen pour la Recherche Nucléaire) through a sustained collaboration among traditional vendors, peripheral research organizations, and a heterogeneous community of voluntary contributors. The purpose of WR—named for the character in *Alice's Adventures in Wonderland* who carries a pocket watch and mutters, "I shall be too late!"—is time synchronization across geographically distributed computing networks. WR consists of a fully deterministic Ethernet-based technology and is currently the clock and event distribution system for CERN's particle accelerators, where time accuracy at the nanosecond level is required.[5] After its implementation at CERN, WR was adopted by other scientific research infrastructures and subsequently implemented in various industrial settings where a common metric of time accuracy across large networks is critical, including high-frequency trading, matching engines in financial services, telecommunications networks, automated vehicles, navigation systems for air traffic control, and smart energy grids.

Based on our findings, we offer three central contributions. First, our work advances an empirically developed understanding of what we conceptualize as the *malleability* of hybrids as a salient characteristic of OSH objects. We adopted the term from physics to allude to the facility of matter to deform under compressive forces. As a metaphor, malleability describes the possibility of changing, modifying, or extending a hybrid object. For our analysis, we define malleability as the propensity of a technology object to be adjusted, adapted, or reconfigured, while (1) fulfilling the intended functionality with similar methods, and (2) retaining salient characteristics of the original technology. This delimits malleable technologies in the same class from alternative technologies that fulfill the same function with completely different means, mechanisms, or methods.[6] Specifically, we argue that malleability is determined by the degree of three salient component attributes over time: embodiment, granularity, and modularity.

[2] See a comprehensive list of examples of OSH at https://www.ohwr.org

[3] https://www.arduino.cc; https://www.reprap.org; https://www.opencompute.org; https://www riscv.org

[4] We employ the term "object" in the same spirit as Faulkner and Runde (2009, 2013, 2019) and Kallinikos et al., (2013) to designate purposefully engineered objects rather than any object that occurs naturally.

[5] A nanosecond (ns) is an SI unit of time equal to one billionth of a second; that is, 1/1,000,000,000th of a second, or 10^{-9} seconds.

[6] As a simple example, floppy disks, CD-ROMs, and solid-state memory all store bitstrings. Yet their core mechanisms of storage are fundamentally different (i.e.,

Second, our paper offers a grounded basis for theorizing about how malleability is a primary determinant of a hybrid development model (i.e., whether is developed employing open source or rather traditional hybrids development model). We reveal how hybrid malleability can vary over time, leading to a process of hybrid *liquification*—which occurs when malleable hybrids are implemented in diverse operating contexts employing open source development methods—or *crystallization*—the integration of malleable hybrids into restrictive legacy systems employing traditional hybrid developmental methods. The processes of liquification and crystallization of a malleable hybrid result from: (1) hybrid maturity (i.e., how advanced it is in the development process), and (2) the interaction with the operational requirements of its physical instantiation (i.e., exogenous forces). Our analysis contributes to an understanding of the interaction of the physical and digital essences of hybrid objects, which is largely underserved in IS research (Ekbia, 2009; Faulkner & Runde, 2009, 2013, 2019; Kallinikos et al., 2013; Yoo 2010; Yoo et al., 2010). Specifically, our theoretical insights enable a more nuanced perspective of the role of a hybrid's material embodiment, salient in the early skepticism of OSH, and describes a rather complex portrayal of how the evolving nature of a hybrid conditions its amenability to open source development (Benkler, 2002; Crowston & Howison, 2006; Feller & Fitzgerald, 2001).

Finally, our work generates policy and managerial guidance on how the potential of OSH can be leveraged if we acknowledge the evolving nature of hybrids. Based on our research, we argue that the traditional separation between hardware and software that often defines how developmental work is organized can be replaced with a more subtle differentiation between less—or more—malleable objects. This insight can help relax the constraints typically perceived in OSH to expand its potential in more ambitious industrial and scientific endeavors.

We proceed as follows. The next section reviews recent conceptualizations of digital objects to delineate the attributes of hybrids and of WR in particular. Then, to position our case and analytical methods, we review how hybrids have traditionally been developed. Third, we review open source literature to underscore the premises of how work is normally organized in open source development. We then turn to our case study of WR. Based on our findings from the analysis of the data, we theorize and formulate a set of key propositions, derive theoretical and normative implications, and conclude with the limitations of the present study and prospects for future research.

magnetism, physical groves on a substrate [lands and pits], or electrons, respectively). Hence, none of these would be considered malleable versions of the other, as their function is completed by fundamentally different means.

Theoretical Underpinnings

OSH as Hybrid Digital Objects

The universe has all types of objects. Our study, however, does not consider all naturally occurring objects. Rather, we adopt the concepts from Faulkner and Runde (2009, 2013, 2019) and Kallinikos et al. (2013) to designate purposefully engineered objects. Objects are entities that endure: "something that exists through time and is fully present at each and every point in time over the period of its existence" (Faulkner & Runde, 2019, p. 5) as continuants (as opposed to occurrents). Additionally, objects are *structured*, made up of *components*: "a number of distinct parts that are organized or arranged in some way" (Faulkner & Runde, 2019, p. 6). Objects possess different *attributes,* which are defining properties based on how the components work, how they are arranged, and how they interact with one another. IS scholars have devoted attention to understanding the specific attributes of *digital objects* (or "digital artifacts") that separate them from *physical objects* (Paavola & Miettinen, 2019).

Physical objects are tangible objects (matter) that can be touched and possess physical substance (Paavola & Miettinen, 2019), possessing spatial attributes such as shape, volume, mass, and location where this physicality is manifested (Faulkner & Runde, 2013, 2019; Leonardi, 2010).[7] Alternatively, *digital objects* have been characterized with attributes such as nonrivalry, infinite expansibility, reproducibility (Faulkner & Runde, 2009, 2013), unboundedness (Ekbia, 2009), interactiveness, fluidity, editability, and distribution (Kallinikos et al., 2010, 2013; Manovich, 2001). From our review of the literature, three salient attributes are relevant to our analysis of hybrids (Table 1). When taken together, they describe a great deal about objects' composition and components, how they are arranged and relate to one another, and their degree of coupling: specifically, their *embodiment*, *modularity,* and *granularity* (Faulkner & Runde, 2013, 2019; Kallinikos et al., 2010, 2013; von Briel et al., 2018; Yoo et al., 2010).

Digital and physical objects are combined in *hybrids* (Faulkner & Runde, 2019). However, this literature stream is less attentive to the physical nature of components (e.g., Ekbia, 2009; Faulkner & Runde, 2009, 2019; Kallinikos et al., 2013; Kallinikos & Mariátegui, 2011). Ontologically, whereas hybrids are often viewed with a separation between digital and physical layers, they receive less consideration of their distinct properties as a unified entity (Faulkner & Runde, 2013, 2019; von Briel et al., 2018; Yoo, 2010). Accordingly, we argue that the category of hybrids warrants its own

[7] There are many manifestations of physical phenomena that that lie outside of this definition in a pure sense (e.g., electromagnetic radiation). For this discussion, we limit our definition of physical objects to tangible objects.

characterization as a class of objects. By treating hybrids as unified entities and examining the interactions between physical and digital elements as they evolve through time, we can obtain insight into a hybrid's unique nature and how this affects OSH development.

A Hybrid Object: White Rabbit

Motivated by the need to eliminate the minute distortions in time measurement across their geographically dispersed particle accelerator and computing network, CERN initiated WR as a complete technology stack of deterministic Ethernet-based technology for time synchronization that uses the IEEE 1588-Precision Time Protocol to reconcile time and phase measurements between a master reference clock and boundary clocks. The two main components in the synchronization hierarchy described by WR are the switch[8] and the node.[9] Both switch and node have hardware, gateware,[10] and software layers, which qualify WR as a hybrid object (Figure 1 and Appendix A). Table 2 describes the components of WR and hardware, gateware, and software layers. Figure 2 illustrates the reference design (the source) of a WR node (a WR component), and Figure 3 shows WR when performing temperature tests.

Table 1. Attributes of Physical and Digital Objects

Attributes	Description
(1) Embodiment	*Embodiment* refers to the component's physical or non-physical state (Faulkner & Runde, 2009, 2019; Yoo, 2010). The notion of embodiment means the property of being manifest in and of the everyday world (Dourish, 2001; Yoo, 2010). Objects with "perpetual embodiment exist in a physical state" (von Briel et al., 2018, p. 281). Physical objects have a *physical* state: "The physical (obviously) can be touched while the conceptual cannot. The material properties of physical objects offer certain opportunities and constraints that simply cannot be overcome—you cannot see through wood or light glass on fire" (Leonardi, 2010, p.1). Digital objects are objects with component parts that include one or more bitstrings (Faulkner & Runde, 2019, p.10). Digital objects with physical components—that is, with a perpetual embodiment (Yoo, 2010)—are hybrids, which "are necessarily material objects, with the physical mode of being of their material components" (Faulkner & Runde, 2019, p.6).
(2) Modularity	*Modularity* is an attribute of object components that determines their coupling (von Briel et al., 2018; Kallinikos et al., 2010, 2013; Manovich 2001; Yoo et al., 2010). The modularity of physical objects describes the relationship between the physical units and defines the relative attribute of an object's structure as opposed to an integral structure. Similarly, the modularity of digital objects' components is an attribute that determines whether components are "responsive to and distinct from each other (e.g., separated by module or layer boundaries) are loosely coupled, whereas components that are responsive to but not distinct from each other (e.g., integrated in one module) are tightly coupled" (von Briel et al., 2018, p. 281). Also hybrids, being simultaneously a physical and a digital object, can similarly possess loosely coupled or tightly coupled digital and physical components, yet it remains unclear how these digital and physical come together describing a more or less modular structure.
(3) Granularity	*Granularity* is an attribute referring to the ability of an object to be decomposed into numerous, small-grained components. Whereas modularity refers to the relationship between components, granularity "entails the stuff of which these blocks are made" (Kallinikos et al., 2013, p. 360), referring to the number of units into which the object can be decomposed (Ekbia, 2009; Kallinikos & Mariátegui, 2011). Physical objects have been qualified as seldom granular because "they are made of blocks or elements thus bundled as to be not readily decomposable and traceable down to elementary units" (Kallinikos et al., 2013, p. 360). In contrast, granularity in digital objects "derives from their ultimately numerical constitution and the ability this furnishes for tracing composite units deep down to the most minute elements and operations by which they are made" (Kallinikos, 2013, p. 360; Manovich, 2001). In hybrids, it remains unexplored how digital and physical components interact, describing a more or less granular hybrid configuration.

[8] A switch is a PTP instance, an 18-port device (boundary clock), that may serve as the source of time for other PTP instances (master) and can synchronize other boundary and ordinary clocks (slaves).

[9] The nodes are single port devices (ordinary clocks) that distribute clock signals, which are physical signals characterized by frequency and phase. WR switches and nodes are interconnected through optic fiber.

[10] Gateware refers to embedded and dedicated code deployed on a field-programmable gate array (FPGA), which is a hardware circuit programmed to implement different logical operations. See Appendix B for a technical note about WR.

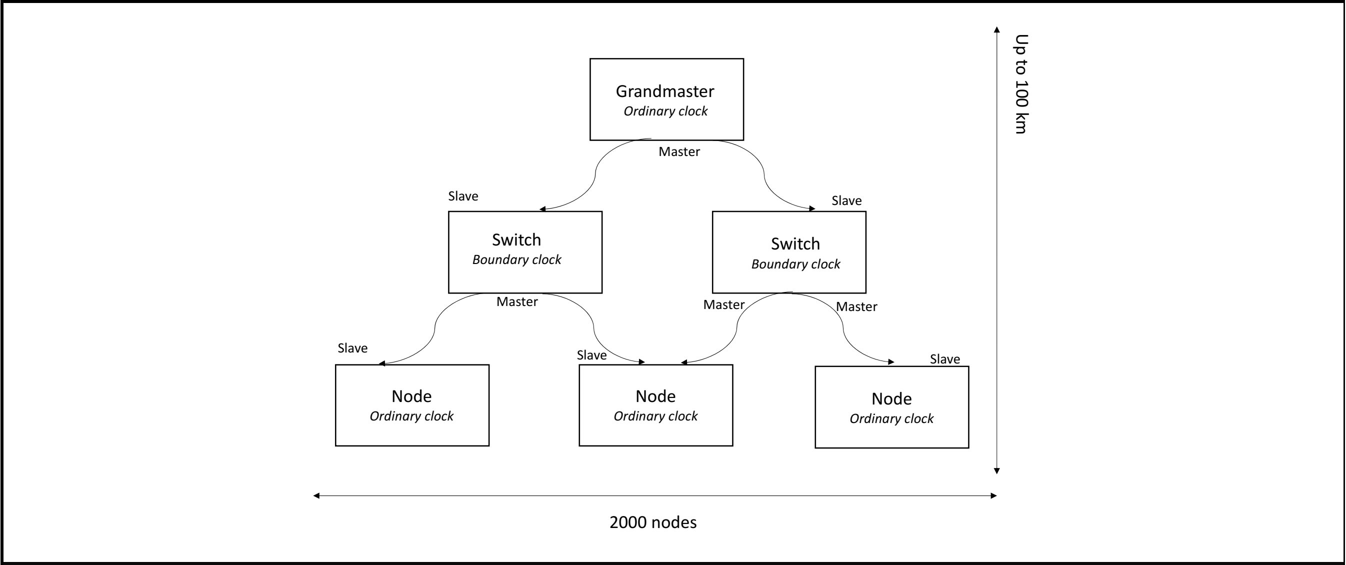


Figure 1. WR Representation (Adapted from Moreira et al., 2009)

Table 2. Decomposing WR Hybrid into its Components

WR main components		**Type**	**Description**
Switch	Physical	Hardware	WR switch box
	Digital	Gateware	General-purpose and dedicated switch gateware—IP cores used both in the switch and interfaces
		Software	General purpose and dedicated switch software
Node	Physical	Hardware	WR Precision time Protocol core and Small Form-factor Pluggable
	Digital	Gateware	General-purpose and dedicated gateware—IP cores used both in the node
		Software	General-purpose and dedicated software—used both in the switch and node

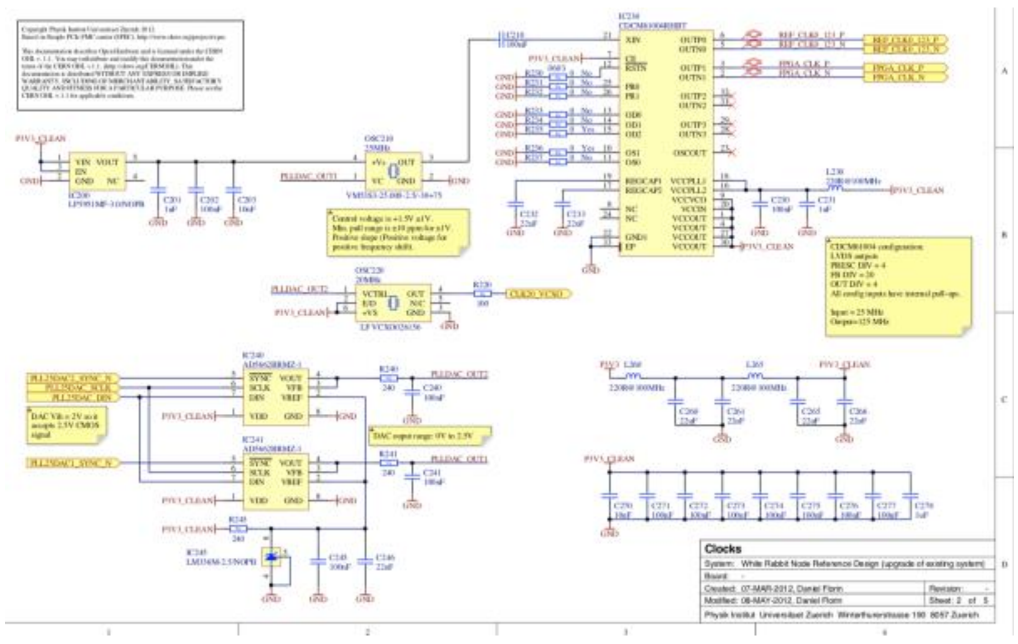

Clocks
System: White Rabbit Node Reference Design (upgrade of existing system)
Board: -
Created: 07-MAR-2012, Daniel Florin
Revision: -
Modified: 08-MAY-2012, Daniel Florin
Sheet: 2 of 5
Physik Institut Universitaet Zuerich Winterthurerstrasse 190 8057 Zuerich

Figure 2. Example of Documentation (the Source) about the Node Reference Design from WR Repository

Figure 3. Temperature Tests of WR (Lipiński et al., 2011)

Traditional Hybrid Development

Although there is no single approach to hybrid development, our purpose is to identify the commonalities across the literature. These include: (1) the logical design of hybrids (i.e., representation of the object), (2) the organization of the development work, and (3) the physical instantiations (i.e., implementations) of the hybrid in the different contexts over time.

The first step in the development of hybrids is the logical design, which is represented in the schematic diagram. The schematic diagram does not provide information on the physical arrangement or interconnection of the parts; it is only a logical depiction of the object. Although one could argue that hybrids and pure software design are similar up to this point, they diverge from here. In the course of their development, hybrids require a translational action to go from the semantic representation of the object to the object itself. Translational action refers to "practices associated with movement from one layer of the bearer to another" (Faulkner & Runde, 2019, p. 10). Moving from the schematic to the actual physical layout is something of an art form, as the physical nature and interconnection of the components (size, heat, etc.) must be considered (Ackerman, 2008). Electronic design automation (EDA) can generate a netlist from component libraries that describes each set of electrical connections by grouping them into a "net," which is a group of components that are electrically tied together. However, despite the benefits of EDA software, substantial human expertise must evaluate the challenges of size constraints, heat, radio interference, external connections, market standards, component cost, and other operational and environmental factors (Drechsler & Breiter, 2007; DeMicheli & Sami, 2013; Gajski & Vahid, 1995). As such, two equally qualified designers could easily produce two circuit boards of varying quality based on the same schematic (Ackerman, 2008).

Several observations about the attributes of traditional hybrid development described in the literature are worth noting and are summarized in Table 3. Traditionally, these development processes have been sequential and have been centrally coordinated with the extensive use of commercial contracts dictating the allocation, interaction, and monitoring of the technology development when conducted across organizations (Cusumano, 1992; Sanchez & Mahoney, 1996; von Hippel & von Krogh, 2003).

Attributes of Open Source Development

Open source development has been characterized as highly voluntary, loosely centralized with parallel collaborations (Dahlander & O'Mahony, 2010; Feller & Fitzgerald, 2000, 2001; Fitzgerald, 2006; Howison & Crowston, 2014; Lindberg et al., 2016; Shah, 2005, 2006). "Open source" is an expression employed to describe both the legal status and developmental model of a digital object, the vast majority of which is software. The legal status of open source digital objects requires that the source code must be redistributable and available to the user, and the creation of derivative work must be permitted under a license that does not discriminate against any user or restrict aggregations of software (Feller & Fitzgerald, 2000). In addition to describing *what* objects can be considered open source, scholars have also identified *how* open source development is organized. Expressions such as "the open source way" of development (Crowston & Howison, 2006) have emerged to describe these common characteristics of how open source objects are developed. We adopt the definition of *development* as the social process of designing, building, and implementing the technical artifact, usually in a specific organizational context and over time (Akhlaghpour et al., 2013).

Table 3. Attributes of Open source and Traditional Approaches to Hybrid Development

Attributes	Open source development	Traditional hybrid development	Problem
(1) Self-assignmentvs. control	**Autonomy and self-selection of tasks:** Open source is characterized by a collaborative effort where agents combine effort voluntarily and self-select their tasks, which does *not* mean that they do not receive pecuniary compensation (though that may often be true), but rather that the collaborators choose their tasks autonomously (Crowston, 1997; Crowston & Howison, 2006; Feller & Fitzgerald, 2000, 2001; Howison & Crowston, 2014).	**Control over the assignment of development activities:** The imperfect translational action from logical design to the object itself requires human expertise. This, combined with the long and interdependent development cycle of hybrids, has traditionally required control over development contributors across organizations (Ackerman, 2008; von Briel et al., 2018; Yu et al., 2018).	For hybrids with a significant HW component, the slower cycle of design/prototype/test requires more purposeful direction giving. Furthermore, technologies of high sophistication require specialized expertise at specific time points. *This renders autonomy and self-selection impractical for certain development phases.*
(2) Loosely centralized vs. centralized	**Loosely centralized:** Open source is characterized by distributed teams who have access to the source code, submit code patches to solve problems, and add functionalities to the software. Open source communities are geographically distributed and remain open and fluid to the entry and exit of contributors. Users can not only contribute to the source code but can also test the software, report bugs, and suggest new features (Feller & Fitzgerald, 2000; O'Mahony & Ferraro, 2007).	**Centralized direction-giving:** Hybrid development draws long development cycles that require centralized direction during their development (Ackerman, 2008). Even if employing virtual prototypes (Bogers & Horst, 2014) or advanced manufacturing techniques like 3D printing, the development of hybrids "involves more activities such as transferring premature prototypes into designs that can actually be manufactured" (von Briel et al., 2018, p. 283). These require more time than modifications to software based on writing lines of code. The inflexibility of engineering modifications later in the process imposes some stability on the core structure. Testing, therefore, is often performed by the engineers who design the object given their integrated nature (Drechsler & Breiter, 2007; Gajski & Vahid, 1995; Mellis & Buechley, 2012; Pan et al., 2018).	The long and interdependent nature of the development cycles and testing activities can necessitate constant coordination. *This can make loosely centralized development impractical.*
(3) Parallel vs. sequential	**Massive parallel development and debugging; asynchronous collaboration and open superposition of tasks:** Open source is characterized by massive parallel development, debugging, and asynchronous collaboration, supported by the internet and concurrent versioning software as a collaborative platform. Discrete development tasks can be completed independently of any required sequence of development. Modules with distinct functionality and payoffs can be isolated and completed. Problematic tasks can be postponed without consequence on future work (Crowston & Howison, 2006; Lindberg et al., 2016; Markus, 2007; Shah, 2006; Shaikh & Vaast, 2016).	**Sequential development activities:** Hybrid development follows discrete, sequential, and interdependent steps. Changes in the fundamental design are more difficult and expensive to modify later in the development cycle, as a change of one component "is likely to require extensive compensating changes in the designs of many interrelated components" (Sanchez & Mahoney, 1996, p. 65; DeMicheli & Sami, 2013). Common software tools only partially alleviate the challenges of tracking and integrating concurrent modifications introduced by different developers (Mellis & Buechley, 2012).	The nature of the development activities, where design decisions need to be agreed upon because they affect the following development steps and any change in the process, requires vast numbers of compensating activities and a very structured, sequential development process. *Parallel development is not an option for certain development phases.*

The basic attributes of open source development have been articulated by its advocates in a large number of publications (Cook, 2001; Masum, 2000; Raymond, 1999) and through diverse case studies (Mockus et al., 2002; Scacchi et al., 2006). Essentially, open source is considered an alternative organizational model for development that is neither market-based nor hierarchical (Shah, 2006). Diverse and partially overlapping approaches have described it as a commons-based peer production (Benkler, 2006); a community-based model (Shah, 2005, 2006); open sourcing (Ågerfalk & Fitzgerald, 2008); collective invention (Allen, 1983); private-collective innovation (von Hippel & von Krogh, 2003); and distributed innovation (Lakhani & von Hippel, 2004).Although open source is not a homogeneous approach to software development, we describe its most frequently mentioned characteristics (i.e., its attributes) in Table 3. As a qualification, it should be recognized that the growing engagement of different commercial interests in open source software development (e.g., Barrett et al., 2013; Deodhar et al., 2012; Fitzgerald, 2006; Spaeth et al., 2014; von Krogh et al., 2012) has relaxed some of the traditional characterizations of the development process (Dahlander & O'Mahony, 2010; Fitzgerald, 2006), where planning and purposive strategies are combined with globally distributed and voluntary contributions (Dahlander & O'Mahony, 2010; Fitzgerald, 2006; Shaikh & Vaast, 2016). Table 3 also specifies a number of challenges that can arise when trying to impose open source development models upon the development of hybrids.

Research Context and Method[redacted]

We engaged in an in-depth single-case qualitative study to generate theory based on the empirical insights. Single-case studies permit a deep understanding of digital objects and of the organizational actions related to their use and development (Klein & Myers, 1999; Orlikowski & Iacono, 2001). In particular, revelatory cases (Gerring, 2007) are useful for theory development. We consider WR a revelatory case in at least two aspects. First, the case offered "an opportunity to observe and analyze a phenomenon previously inaccessible to scientific investigation" (Yin, 2003, p. 42). Access to the CERN Hardware and Timing Section at the Beams Department that coordinated WR development offered us the opportunity to obtain a decade of rich longitudinal data on the development of such a sophisticated technology that has been unprecedented in OSH development (Bonvoisin et al., 2017; Boujut et al., 2019). Second, WR is an example of OSH development that relied on a diverse community of technical specialists, commercial vendors, industrial complementors, and voluntary contributions, as a collaborative development endeavor unparalleled in OSH development (Pearce, 2017).

Research Context

WR is the name of an OSH initiated in 2008 when engineers at CERN were confronted with limited bandwidth and the impossibility of dynamically evaluating the delay induced by the data networks that constitute CERN's geographically distributed computing infrastructure supporting the world's most powerful particle accelerators. WR was developed with the following unprecedented specifications: (1) the transfer of a time reference from a central location to many destinations with an accuracy better than one nanosecond and a precision[11] better than 50 picoseconds,[12] (2) the ability to service more than 2,000 nodes, (3) the ability to cover distances in the order of 10 km (although it achieved distances over 100 km in its development process), and (4) data transfer from a central controller to many nodes with a guaranteed upper bound in latency.

Prior to WR, the extant synchronization standard for Ethernet networks was the precise time protocol (PTP), which is standardized as IEEE 1588. WR extends PTP in a backwardly compatible way to achieve sub-nanosecond accuracy (Moreira et al., 2009). "The combination of deterministic latencies with a common notion of time to within one nanosecond allows WR to be a suitable technology to solve diverse problems in distributed real-time control and data acquisition" (Lipiński et al., 2011, p. 2).

WR started in 2008 as an OSH when CERN decided to collaboratively develop the technology with any volunteer contributor, publishing an open call in CERN's vendor network, supported by a repository, Wiki, developers' mailing list, workshops, and a set of collaborative tools. Most importantly, an open source hardware license was created to govern the rules of sharing, distributing, and selling the WR designs. Very early on, the GSI-Helmholtz Centre for Heavy Ion Research (GSI), a large particle accelerator facility in Germany, joined. Soon a progressively larger group of organizations engaged to shape a diverse and vibrant community that contributed to WR development. The number of contributors has grown beyond any initial expectation and has surpassed CERN's ability to keep track

[11] Accuracy refers to the proximity of measurement results to the true value, whereas precision is the degree to which repeated (or reproducible) measurements under unchanged conditions show the same result.

[12] A picosecond is an SI unit of time equal to 10^{-12} or 1/1,000,000,000,000th (one trillionth) of a second.

of the different WR reuses and adaptations. In less than a decade it had proliferated into a "multilaboratory, multicompany and multinational collaboration developing a technology that is commercially available, used worldwide, and incorporated into the original PTP" (Lipiński et al., 2011, p. 2).

Data Collection and Sources

Our primary sources of data were 38 semi-structured and open-ended interviews that we conducted with selected WR community actors. These actors included WR developers from research infrastructures; companies contributing to WR development; and WR users implementing the technology, reporting bugs, and helping improve WR. In addition to the interviews, we conducted direct observations during two study visits to CERN, in April 2017 and October 2018, including attendance at a WR developer workshop with more than 56 participants.

The interviews were chosen on the initial recommendation of the WR lead team at CERN, with subsequent recommendations from the interviewees. Our objective was to interview a representative cross-section of the WR community to broadly understand the subject area of hybrids and OSH development and further enable an inductive generation of theory from the empirical data. Three primary themes guided the interviews: (1) what is WR's functionality and structure, and how does it work? (2) How did different organizations become involved in WR development? and (3) What is the WR development process and how is work organized across contributors? Whereas these three major themes guided the initial interviews, we allowed for open-ended discussions around WR to obtain a deeper understanding of the technology and different aspects of the development process. We subsequently conducted three additional rounds of interviews, where the protocols evolved toward more detailed and focused topics that emerged from the previous iterations.

The last stage of data collection was devoted to the confirmation of our interpretations and refining of WR attributes and relationships with the development characteristics. Appendix B describes the four stages of data collection and illustrates the interview protocol used in the later phases in Table B1 in Appendix B. The interviews were conducted in English and Spanish from October 2017 to September 2020. Each interview was between 20 min and 90 min long. Table B2 in the Appendix B provides details on the study's primary data and the alphanumeric key identifiers, representing quoted interviewees.

A second important source of data was the information retrieved from the WR repository and Wiki, which contains general information about the WR project and technology, WR users, and the open hardware license. We also gained access to the standard working group IEEE1588-2008, where WR was standardized, to reference the technical documentation describing WR structure for our analysis. We employed these secondary sources to obtain a deep understanding of the technology and the development processes documented by the developer community (see description in Table 4). Secondary sources were also employed to corroborate evidence from primary data. Table B3 in Appendix B illustrates the use of primary and secondary sources in our empirical analysis of WR development.

Data Analysis

We performed a four-stage data analysis by relying on established procedures for inductive research (Gioia et al., 2013; Miles & Huberman, 1994).

First, our analysis methods involved a detailed study and reflection of multiple textual materials and abundant information available online about WR. We produced brief summaries that moved from technical descriptions to managerial inferences. Publications related to WR and the large volume of information available about the technology helped us to have more technically focused conversations with informants active in both the WR technology stack and the OSH developer world (Lok & de Rond, 2012).

Second, we iteratively analyzed the interview transcripts by coding relevant observations and contrasting them with our analysis of secondary sources. In the open coding procedure, we coded at various levels to delineate the main concepts in the empirical data and generated research memos that synthesized the emergent themes. During the open coding, we broke down our data to understand WR attributes and the characteristics of the development processes. These memos progressed into extended notes where we consolidated repetitions and gradually collapsed our codes into first-order categories (Gioia et al., 2013). During this process, we periodically discussed any discrepancy in the interpretation and went back to the empirical data whenever necessary. For instance, in this process, we began dissecting the WR object and its structure (i.e., the switch, node, and other components) and capturing how the WR community described and qualified WR technology. We also leveraged the empirical descriptions of how WR component designs evolved and how the developer community described the processes for developing WR, and we identified the diverse WR implementations across different operating contexts.

Table 4. Summary of Secondary Data Collected

Types of data	Description	Use in the analysis
Repository	• 5,076 commits • 36 developer members	To gather data and obtain an overall understanding of all WR technology, its components, cycles of development, different component versions, meetings among contributors, and main events in WR development.
Wiki	• Documentation about: • WR technology: WR switch, master (data, timing), node (WR PTP core), WR good practice guide, calibration (default parameters for WR switches/nodes, procedure), data delivery, synchronization, standardization in IEEE1588-2008, and a frequently asked questions section. • WR users: 30 users of WR and 16 evaluating the technology (documentation about the organizations, descriptions, and presentations) • WR projects: 13 publicly funded projects using WR	
Newsletters	5 newsletters (2013, 2014, 2015, 2018)	
Meeting minutes published	10 meeting minutes (2008-2018)	
Workshop	• 10 workshops (2008-2018) • 1 developer meeting (2010) • 2 tutorial WR workshops (2017, 2018)	
Blogs/websites	43 websites of users and projects	
Publications	• Presentations (n = 64) • Papers (n = 53) • Master thesis on WR (n = 2) • Posters (n = 2) • Demos (n = 3 in 2010 and 2013) • Training material (n = 2 in 2013 and 2016) • Test reports (n = 18)	
Standardisation documentation from IEEE 1588 Working Group	• Working draft on IEEE 1588 standardization 2018 • Technical information provided in the shared group IEEE 1588 standardization	

Third, in the process of axial coding, we structured our first-order categories into second-order themes and higher-level aggregate dimensions (Gioia et al., 2013). Within this process, we gradually progressed toward a more theory-driven explanation. We performed this process repeatedly, making extensive use of notes and observations to interpret the data. For instance, the construct of malleability and the identified processes of liquification and crystallization were consolidated in this analytical step. Malleability emerged from aggregating the qualifications of WR technology across the second-order themes such as modularity, granularity, and embodiment, and codes such as "rigid," "compact," and "unbreakable," from the transcript's analysis. Liquification emerged from integrating second-order codes describing WR development across the logical design, diverse physical instantiations in scientific settings, and descriptions such as: "and then people started taking WR and adapting it to different formats," "without us doing it"; "however, the fundamentals of the protocol are the same." Similarly, crystallization surfaced from integrating second-order codes describing WR development and was revealed from codes such as "proprietary implementations," "to meet the requirement of picosecond accuracy in," and "most (industrial) applications are concerned with performance" from the transcript's analysis. Figure 2 presents the data structure resulting from this phase.

Finally, we focused on disentangling the linkages between our dimensions to build a cohesive model theoretically explaining how the attributes of the hybrid affected the development model. At this stage, we integrated all major concepts from hybrid malleability with the development characteristics to form a holistic and coherent theoretical model. Once the core categories and the model emerged, we contrasted them with prior literature on digital objects and OS development (see the Discussion section) (Bryant &

Charmaz, 2007). We applied respondent validation (Miles & Huberman, 1994) by sharing our initial findings with the participants of the study. To gather feedback about our study, preliminary results were presented at a workshop of the WR community on October 6-7, 2018.[13] Additionally, editable tables about the structure and development history of WR were shared with the interviewees so that the community could confirm, correct, or elaborate on our description of the technology and its development process. Figure 4 illustrates the emerging data structure through our analysis, and Table B2 in Appendix B illustrates the progression of our empirical analysis with selected quotes and empirical observations.

Findings

A detailed analysis of our WR case enabled us to discern the structure and evolution of WR. This analysis gave us insight into what we conceptualize as the malleability attribute of hybrids, which we view as a function of three structural attributes of the object: (1) its embodiment (i.e., material components), (2) granularity (i.e., the decomposability of the components), and (3) modularity (i.e., whether the components are tightly or loosely coupled). We next describe WR development in three distinct phases that explain how the attributes affecting hybrid malleability changed through time and how these changes affected—and were affected by—the development model of the WR community.

WR Development

Phase 1: Hybrid Formation (2008−2012)

The first *phase* began with the project initiation in 2008 and concluded when the first working version of WR was produced in 2012. WR was launched as an OSH project, a decision that is consistent with CERN's traditional operational philosophy and *raison d'être.* However, it quickly became evident that CERN, as the sponsor and principal user of WR, needed to (1) *centralize* WR development and (2) *control* the assignment of development activities for its major (3) sequential and interdependent steps. As explained by RSE1: "we tried to replicate as much as we can as an open source project but although we tried, the problem we had is the time of iteration and inertia, this was a huge problem and probably the main one."

First, as RSE6 explained, the organization of WR development in this first phase was highly centralized: "There was internal work at CERN and external work by different companies. And all this work was coordinated at CERN and integrated at CERN to make it work together." Second, contractual arrangements were employed by CERN to control the development of the first WR prototype, and this required tight direction and supervision of tasks and development teams. The contractual agreements included: (1) contracts awarded to companies to gather and manage WR specifications across the WR development community; (2) contracts to develop the repository and main hub for WR collaboration; (3) further contractual arrangements to contribute to the first switch and node prototypes; and (4) contracts for prototyping, where manufacturers were asked to produce a few units of WR components and distribute them across the community for testing. All of these contractual arrangements specify that all documentation that results from the development must be shared in the repository and is governed under an open source license. An interesting facet of the contractual agreements was that many vendors included voluntary contributions as part of their deliverables. This implies that, if their component included volunteer contributions from the WR community, they were equally responsible: "You must be ready to document and publish everything. Support may take more than you want" (RSE4). Third, the development followed the four major sequential and consecutive steps of: (1) requirement and specifications, (2) design, (3) prototyping, and (4) testing. The outcomes of each step were highly dependent on the results of the previous step.

The analysis of our data suggests the following reasons for these three development attributes: First, in the words of RSE6, regarding the implications of hybrid embodiment:

> *The development cycle was longer for WR [compared to other open source software]…. in WR the issue is that, for instance, a modification in the PCB [see Table B2] of a relatively complex component like the WR switch can take several months of work so we need to discuss it.*

Second, RSE1 described the constraints of making the hybrid more granular: "Software is easier to split among companies, but that does not make sense for hardware. In WR it did not make any sense. For precision, it is also better that you do not have too many connectors; here, it was not practical." CH1 further explained the low granularity: "you could not make it more granular; it would make it many times more expensive and need extra work. There would be less efficacy in terms of precision. It would be harder to make it work."

[13] The workshop information is published at https://www.ohwr.org/projects/5/wiki/oct2018meeting

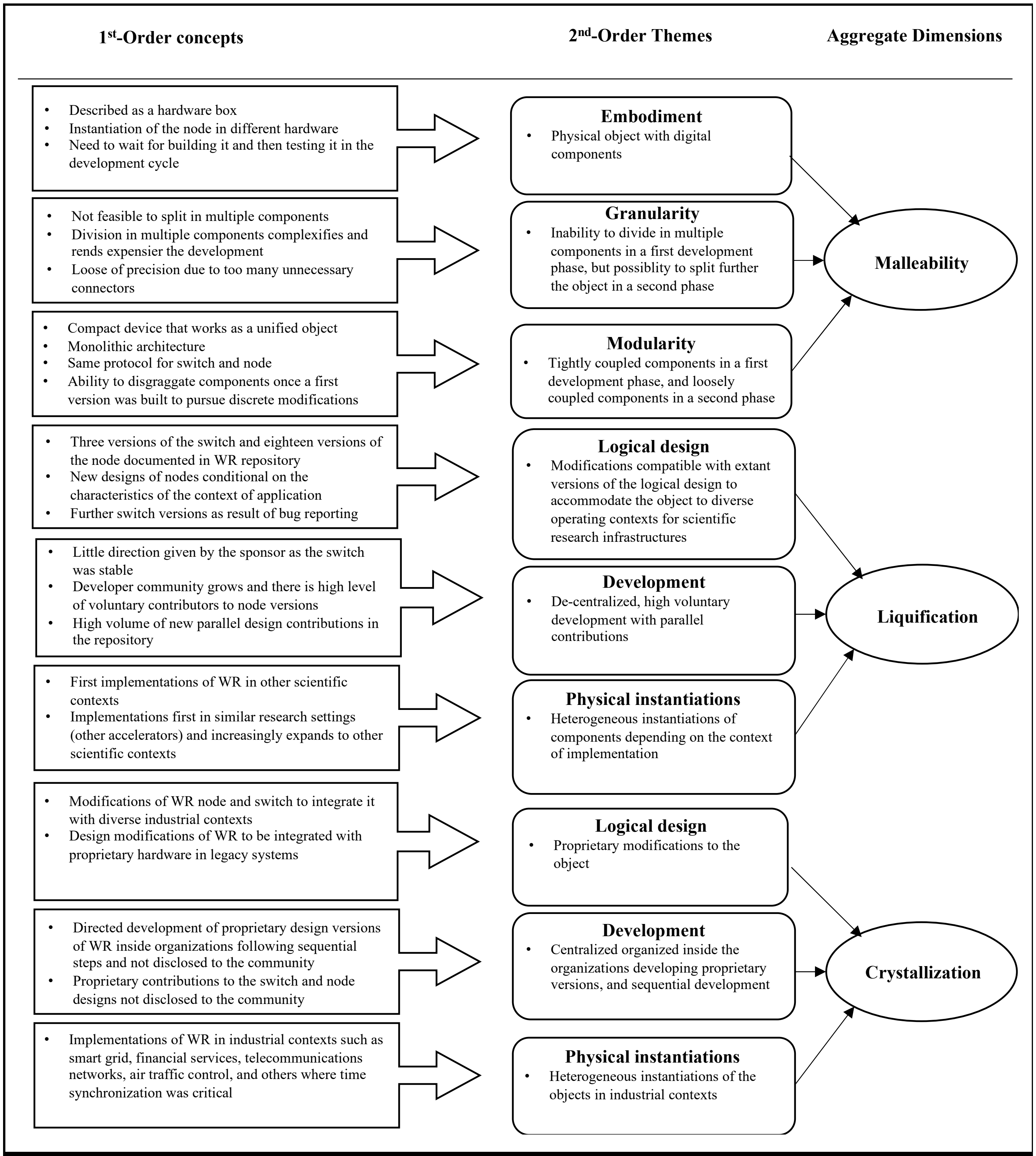

1st-Order concepts
2nd-Order Themes
Aggregate Dimensions
• Described as a hardware box
• Instantiation of the node in different hardware
• Need to wait for building it and then testing it in the development cycle
• Not feasible to split in multiple components
• Division in multiple components complexifies and rends expensier the development
• Loose of precision due to too many unnecessary connectors
• Compact device that works as a unified object
• Monolithic architecture
• Same protocol for switch and node
• Ability to disgraggate components once a first version was built to pursue discrete modifications
• Three versions of the switch and eighteen versions of the node documented in WR repository
• New designs of nodes conditional on the characteristics of the context of application
• Further switch versions as result of bug reporting
• Little direction given by the sponsor as the switch was stable
• Developer community grows and there is high level of voluntary contributors to node versions
• High volume of new parallel design contributions in the repository
• First implementations of WR in other scientific contexts
• Implementations first in similar research settings (other accelerators) and increasingly expands to other scientific contexts
• Modifications of WR node and switch to integrate it with diverse industrial contexts
• Design modifications of WR to be integrated with proprietary hardware in legacy systems
• Directed development of proprietary design versions of WR inside organizations following sequential steps and not disclosed to the community
• Proprietary contributions to the switch and node designs not disclosed to the community
• Implementations of WR in industrial contexts such as smart grid, financial services, telecommunications networks, air traffic control, and others where time synchronization was critical
Embodiment
• Physical object with digital components
Granularity
• Inability to divide in multiple components in a first development phase, but possiblity to split further the object in a second phase
Modularity
• Tightly coupled components in a first development phase, and loosely coupled components in a second phase
Logical design
• Modifications compatible with extant versions of the logical design to accommodate the object to diverse operating contexts for scientific research infrastructures
Development
• De-centralized, high voluntary development with parallel contributions
Physical instantiations
• Heterogeneous instantiations of components depending on the context of implementation
Logical design
• Proprietary modifications to the object
Development
• Centralized organized inside the organizations developing proprietary versions, and sequential development
Physical instantiations
• Heterogeneous instantiations of the objects in industrial contexts
Malleability
Liquification
Crystallization

Figure 4. Data Structure

Third, the following comment made by respondent RSE6 captures the low modularity of WR; that is, the high coupling between switch and node and the component layers: “For WR, we think of hardware boxes.” As RSE1 added:

> *The switch is quite a compact device; it must work like one unified device, and if you have different companies, you need to define different interfaces between the parts that they are designing, test each part, see that they work together, and you make it much more complex, too much work, and much more expensive.*

In sum, although some asynchronous development was conducted by voluntary developers from peripheral research organizations, in this first phase, WR displays *low malleability*. The design of the core WR technologies with exacting performance requirements on a monolithic hybrid architecture resulted in physical embodiment, low granularity, and low modularity, which lead to a predominance of practices from traditional hybrid development.

Phase 2: Hybrid Development in Local Contexts (2012−2015)

The *second phase* began with the first WR prototype release. At this point, the first users began implementing WR and reporting bugs, the fixes of which were incorporated into further designs. Novel instantiations of the node began to appear based on the unique requirements of the installations of other scientific research infrastructures. Extraordinary examples include: (1) meteorology research institutes that must transfer time from atomic clocks over distances up to 1,000 km; (2) the neutrino telescope KM3Net, located in the deepest seas of the Mediterranean; and (3) the world’s largest and most sensitive cosmic ray observatory for gamma-ray astronomy, the Large High Altitude Air Shower Observatory (LHAASO), built in China at 4,410 meters above sea level. However, modifications, both physical and programmatic, were often needed: “In embedded detector electronics there is usually not much space available. A standard WR switch is not suited for detectors like Chromium or KM3NeT” (WR repository, 2020).

In this phase, WR development is characterized by being (1) *loosely centralized*, (2) *highly voluntary,* and with (3) *parallel contributions* toward modest changes to the design of the switch, but with many new designs and configurations for the nodes proliferating from a flourishing and increasingly heterogeneous WR community. First, with the switch stable, it was possible to proceed with *loosely centralized* control, permitting many new designs of the nodes:

> *What happened is that we designed one node, which is the spec card, and then people started taking it and adapting it to different formats ... one company developed a simplified version. Some people took this design and made different formats, and this was without us doing it, we did not pay for the design, it was because people needed it* (RSE6).

Second, in the words of RSE1, regarding one of the *highly voluntary* contributions:

> *There was an engineer from South Africa who was interested in solving particular problems with much greater temperature variations in South Africa compared to CERN, where optical fibres are underground and naturally isolated from the temperature variations of the atmosphere. So, he discovered effects that we had not seen and voluntarily improved WR in what affects timing for long distances.*

Third, within this phase, a more heterogeneous community of WR users engaged in *parallel developments* to provide alternate versions of WR to customize it to the specific operational requirements of their infrastructures. The LHASSO scientific experiment in China, for instance, developed the so-called “mini-WR node,” which is the smallest WR node created to give flexibility for connector and panel arrangement on the carrier board. Other node designs from scientific organizations included an additional FPGA to increase processing capability, or plug-in versions of the node to allow stand-alone implementations.

The analysis of our data suggests the following reasons for the predominance of loosely centralized, voluntary, and parallel contributions. First, the low granularity and low modularity of the WR switch and node hierarchy, which were highly determinative in the first phase of development, were less of a constraint in later phases. Once the development of the WR stack was sufficiently stable, developers in a second phase could disaggregate WR components—and thereby increase granularity and modularity—to pursue discrete modifications of the node and switch in parallel. As this quote from RSE6 illustrates, where the switch and node were initially developed together, different node configurations and switch improvements were possible in subsequent implementations:

> *What the switch and node have in common is that: what is implemented on each part of the switch is the same on one node, plus the WR protocol, which is the same on the switch and the node. Yet, in simplified words, whereas the node has the wider connection, so the fibre, on the other side is usually used for something. And because of that, you may need various different node versions.*

In summary, the second phase of WR displayed *higher malleability*, due to evolving WR structure (i.e., immutable embodiment with higher granularity and modularity) combined with the increasing maturity of the hybrid, which leads to a predominance of open source development attributes.

Phase 3: Development of Hybrid Derivative Works (2015−2020)

The *third phase* began when WR started a standardization process to guarantee the stability of the technology, which raised awareness about the potential of WR across industries outside of scientific research. In this phase, we found a growing number of increasingly heterogeneous WR implementations by industry. Examples of implementations include Vodafone, which conducted a successful proof of concept in 2017 to distribute accurate timing through the live Vodafone network, where time was measured with a surprisingly small error of less than one nanosecond over a cascade of four sites that spanned a total distance of 320 km. In financial services, the Frankfurt Stock Exchange implemented WR because, as described by CH1:

> *Financial transaction organizations are required by law to prove that the time reference used for stamping transactions is UTC [Universal Time Coordinated] traceable. Thus, the accuracy required is in the millisecond range, but WR enables the nanosecond range with high accuracy and so enables legal timestamping applications.*

In the third phase, although some control was exercised by the sponsor, it was mostly in the further standardization of core technologies and logical design. WR industrial implementors exercised greater autonomy in technology modification in legacy systems. As described by CH6:

> *The way you do an open source on software, you tend to get it to be implemented in different microprocessor architectures. However, hardware is different. So, it's like any time you put two pieces of hardware together, they are never going to be exactly the same. Therefore, as WR evolves, the real challenge is that they have tried hard to make it such that you just push a button and things are configured automatically and pull in the relevant files for architecture, but it does not work like that. It is not easy; this is really hard.*

New versions of WR switches and nodes were developed as proprietary applications, controlled by different companies implementing WR, which were not always disclosed to the WR community.

In this phase, WR maintains the malleability from Phase 2 but is augmented by (1) *centralized* development and high pecuniary control over the assignment of tasks by companies developing WR industrial implementations, (2) *fewer/fragmented voluntary* contributions compared to the previous phase, and (3) *sequential proprietary* developments to improve the design of WR and integrate it in different industrial technologies.

Our data analysis suggests the following reasons: The embodiment remained as it was—but both the high granularity and modularity (easily decoupling of switch-node) salient in the second phase were again reduced in the third phase in some individual implementations. This was due to the fact that many industrial applications required that WR be integrated with legacy systems. A company developer explained the constraints of legacy systems: "What we want to try with proprietary implementations is to meet the requirement of picosecond accuracy, but not with custom hardware as we want to make it more generic—and removing all our infrastructure would not make much sense." Another company developer added: "The main limitation of WR is that it is a brand-new hardware design, a custom hardware. It does not have the off-the-shelf components and we should develop things to overcome these limitations." Some of the improvements in the previous WR switch design included frequency stability in terms of noise reduction, less power consumption, and enhanced monitoring integrated into the data visualization tools, providing information about WR performance in higher-level information systems.

In summary, we have presented the evolution of the WR as occurring in three main phases, which are defined by the nature of the development that transpired, the developmental model employed, and the source and locus of the developmental efforts coming from both the WR sponsor and the external WR community. A critical focus of our interpretation of how WR evolved and matured through time includes the interplay between endogenous attributes of the hybrid, how these evolved intact with developmental maturity and exogenous requirements, and how these modifications further informed the evolution of the WR logical design. Table 5 summarizes these insights, which we subsequently elaborate.

Table 5. Summary of Relationship Between Object Attributes and Development Model

	Phase 1 **(2008-2012)**	**Phase 2** **(2013-2015)**	**Phase 3** **(2016-2020)**
Logical design: How WR objects and components evolved	WR OSH v0.0 ➔ v.1.0 First version of the switch and node controlled by CERN.	WR OSH v1.0 ➔ v3.0 (switch) and 18 versions of the node documented in WR repository. IEEE1588-2008 New designs of WR nodes, which are conditional on the characteristics of the environment where WR is applied. Further switch versions are a result of bug reporting to the switch.	WR OSH v3.0 ➔ v.3.4 ➔ v 4.0 (ongoing) (switch) and multiple proprietary versions of WR. IEEE1588-2019 (PTP high accuracy) Release of different versions of the switch with improvements and extensions made. Proprietary versions of WR: first implementations of WR in other industries (e.g., financial services, telecommunications) lead to new proprietary versions of WR switches and nodes.
Development characteristics: How WR development was organized	**Centralized, control, sequential**: Strong direction provided by the sponsor was given to the design. Contractual agreements with HW and SW suppliers to allow a first prototype to emerge. Few voluntary contributions that include few research infrastructures. **Endogenous attributes:** The object does not allow for massive parallel development, but sequential and highly dependent steps that need to be directed because of *embodiment, low granularity,* and the *tightly coupled* nature of components, which make the structure highly interrelated in particular for precision reasons. **Exogenous requirements:** Prototype developed for particle colliders.	**Decentralized, highly voluntary, parallel contributions:** Little direction given by the sponsor as the switch was stable, while there was a high level of generativity, as the new designs of the nodes were shared in the repository as the WR community was growing. High level of voluntary contributors to design multiple versions of the node. Contractual agreements for WR production. **Endogenous attributes:** The object allows for parallel, generative, and decentralized developments of new design versions of the node with a stable version of the switch because there is *high granularity and modularity* where node-switch development can be decomposed but also between its embodied and nonphysical components once the first prototype is developed. This generates heterogeneous node designs for different WR instantiations. **Exogenous requirements:** Modifications to node (primarily) to accommodate diverse operating contexts for scientific research infrastructures.	**Centralized, less voluntary, sequential**: Proprietary design versions of WR were directed inside the organizations following sequential development processes and not disclosed to the community. Voluntary contributions are balanced by proprietary contributions to the switch and node designs. **Endogenous attributes:** While new versions of the switch and node are pursued, the switch and node need to be bundled again, reflecting a low *granular* and *tightly coupled* structure when the *embodied* object must be integrated with legacy systems for different industrial applications. **Exogenous requirements:** Modifications of node and switch to integrate into legacy systems across diverse industrial applications.
Physical instantiation: WR implementations and visual representation	Hybrid formation: WR prototype for development/proof of concept. Internal at CERN/GSI.	First implementations of WR in other scientific research infrastructures.	Integration of WR in industrial applications.

Key Propositions Emerging from the Analysis

We present a conceptual model (Figure 5) that depicts a set of key propositions emerging from our empirical investigation. Our figure illustrates how the three hybrid attributes (i.e., granularity, modularity, and embodiment) determine the malleability of a hybrid object. This is moderated by the developmental maturity of the hybrid. Where an object with low malleability is amenable to traditional hybrid development methods, high malleability is more suited to OS development modes. Once the core technologies are sufficiently stable, the peripheral layers of the technology stack can be customized to specific implementation contexts in a process of liquification. *Liquification occurs when the core is sufficiently stable to permit modifications of peripheral elements of the hybrid technology stack to accommodate the requirements of diverse (exogenous) implementation contexts yet remain consistent with the logical design of the OSH license.* In extreme contexts where the existing legacy systems are far from the standards of the core technology, new versions of the hybrid can be developed in a process of crystallization. *Crystallization occurs when the core of the hybrid technology is stable, yet exogenous implementation contexts require instantiations that push the technology object outside of the boundaries of the logical design described by the OSH license and thus become discrete instantiations that fulfill the functionality of the original technology yet are incompatible with the OSH license.* We reflect upon and further refine this logic in what follows.

Our analysis departs from platitudes such as "bits are free, atoms are not" that were salient in early skepticism of OSH (Balka, 2011; Balka et al., 2010; Boisseau et al., 2018; Oberloier & Pearce, 2018; West & Kuk, 2014). It offers a more refined understanding of the role of a hybrid's material embodiment and under what conditions a hybrid's malleability makes it amenable to open source development. For hybrid malleability, embodiment is determinative but not so much as a function of postdevelopmental manufacturing cost. Rather, what is important is how embodiment combines appropriate levels of modularity and granularity, resulting from designing the optimal hybrid to fulfill its operating requirements. This leads to Proposition 1.

Proposition 1: *High granularity, high modularity, and nonmaterial embodiment increase a hybrid object's malleability.*

A central insight from our analysis is that malleability is not static. In Phase 1, the primary focus was on the development of the core functionality and performance of the technology by the sponsor for implementations in similar contexts (i.e., particle colliders). The demands emphasized endogenous attributes of the technology, where the tight coupling of the components with limited modularity was required to maximize WR performance. In Phase 2, as WR was implemented in heterogeneous environments with different operating conditions, the exogenous factors of the WR physical instantiations played a determinative role in forcing the decoupling and modification of the node. This suggests that the hybrid is not exclusively defined or constrained by its endogenous attributes. Rather, as the technology matures to permit a decoupling of the hybrid's layers, exogenous requirements of WR instantiations in scientific settings (e.g., volatile temperatures, very high altitude, or deep sea level) further stimulated the evolution of granularity and modularity and thereby increased its malleability. This leads to Proposition 2.

Proposition 2: *The malleability of a hybrid increases with maturity.*

Our analysis suggests that when WR was in a very early stage of development and a proof of concept or working prototype was yet to be produced, the development had to be highly centralized with limited volunteer contributions completed in highly structured and sequential steps. Although WR was officially open source, most external contributions came from a few specific vendors operating with commercial contracts and there were few voluntary contributions. Our data clearly indicate that an alternative development model with numerous components, interfaces, or unanticipated volunteer contributions would not only increase the development efforts of the sponsor but also reduce the operating precision of WR. Consequently, our study suggests that if a hybrid is not sufficiently malleable, a traditional mode of hybrid development occurs, leading to Proposition 3.

Proposition 3: *Less malleable objects are associated with a development model characterized by (a) assigned tasks and pecuniary compensation, (b) centralization, and (c) sequential collaboration.*

Once the core technology was complete and stable, implementations began outside of CERN in other scientific research infrastructures. Whereas some of these were particle accelerators such as GSI, others were completely different and required WR to operate distinctively. As a result, implementors and voluntary organizations were able to decompose the WR stack to modify the node to the specific operating conditions of their installations. This ability to take an object that was once low in malleability and decouple it into its many components makes it more malleable and capable of independent modification by developers working in parallel. Consequently, WR evolved into a more highly malleable object that enabled voluntary contributions where developers self-assigned their development activities in independent, parallel, and asynchronous contributions. Proposition 4 follows.

Proposition 4: *Highly malleable objects are associated with a development model characterized by (a) self-assigned tasks, (b) decentralization, and (c) asynchronous collaboration.*

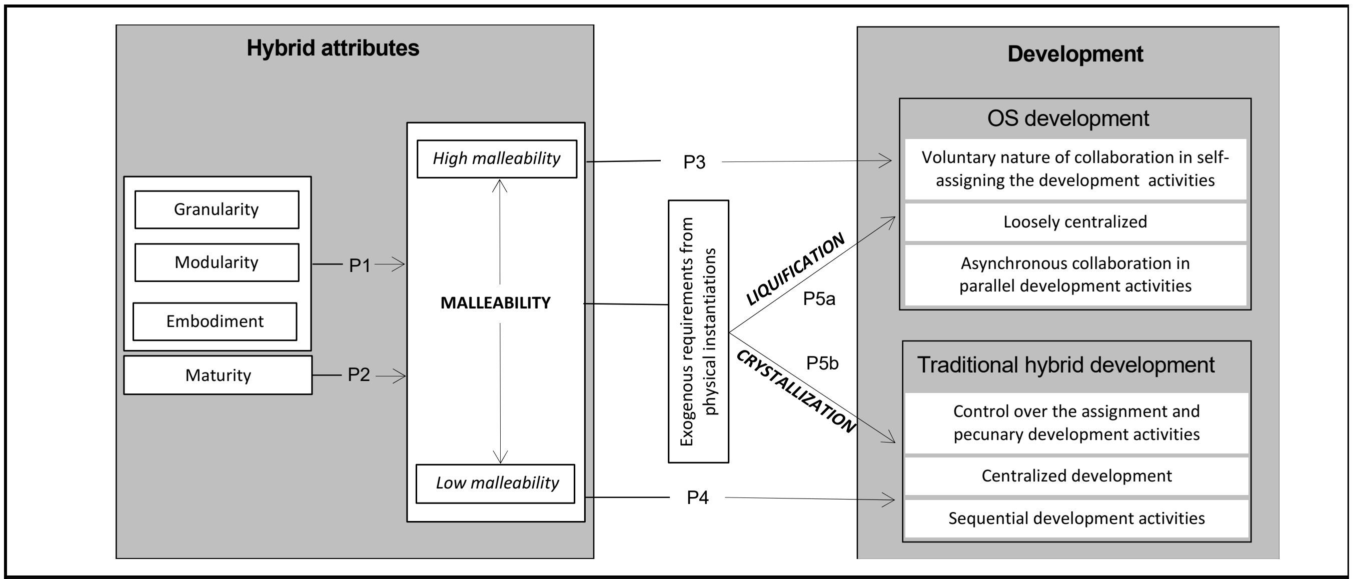

Hybrid attributes
Granularity
Modularity
Embodiment
Maturity
P1
P2
High malleability
MALLEABILITY
Low malleability
P3
P4
Exogenous requirements from physical instantiations
LIQUIFICATION
P5a
P5b
CRYSTALLIZATION
Development
OS development
Voluntary nature of collaboration in self-assigning the development activities
Loosely centralized
Asynchronous collaboration in parallel development activities
Traditional hybrid development
Control over the assignment and pecunary development activities
Centralized development
Sequential development activities

Figure 5. Conceptual Model

Our case data indicates that consistent with Proposition 2, with maturity and higher malleability, numerous independent modifications were made at different research infrastructures. For example, at the end of Phase 2, 18 different versions of the node were documented on the WR repository. At the same time, given the commonalities of the research infrastructures, as well as the fact that they were all compliant with the shared WR logical design, they are considered to be in the same category of interoperable technology. We conceptualize *hybrid liquification* as a metaphor to describe the process by which the hybrid components are constantly modified through open source attributes (i.e., decentralized, highly voluntary, parallel contributions). Multiple variants of the hybrid components result from the liquification process and are *compatible with extant versions of the WR logical design*—that is, heterogeneous node designs for different WR instantiations that are compatible with the standard WR core technology.

In Phase 3, hybrid malleability continued yet resulted in a different process for some physical instantiations. When WR was implemented in increasingly heterogeneous industrial settings, such system integrations often required the hybrid to be modified to the operating specifications of a different sector and, additionally, integrated into legacy systems that could be equally distinct. As is evident from our case data, the nature of the legacy systems often played a constraining role to the extent that the WR technology required substantial modifications to be compatible. In extreme cases, this generated instantiations of WR where the switch-node became recoupled for easier integration and interoperability with legacy systems. Such modifications for WR integration were performed in a highly centralized manner, with control over the assignment of developmental tasks. We term this process *crystallization.* Crystallization is a metaphor that we employ to describe the process by which core technologies are modified, employing traditional hybrid development attributes (i.e., centralized control over sequential development tasks) and resulting in variants *that diverge from the WR logical design* to become proprietary versions of WR. This did not happen with all the WR instantiations in Phase 3. Rather, the liquification and crystallization processes occurred in parallel. This leads to propositions 5a and 5b.

Proposition 5a: *With high malleability, a hybrid can liquify with increasing requirements of diverse operating contexts.*

Proposition 5b: *With high malleability, a hybrid can crystallize with increasing requirements of diverse operating contexts with restrictive legacy systems.*

Our final insight is that it is important to differentiate physical instantiations or implementations of WR across its development from the evolution of the logical design shared openly and regulated by a WR OSH license. The evolution of the WR instantiations followed a path with homogeneous versions developed for particle accelerators in Phase 1, complemented with moderately diverse modifications of the node for distinct research infrastructures in Phase 2 (i.e., liquification), combined with extensive modifications to both the switch and node to integrate into legacy systems in diverse industrial settings in Phase 3 (i.e., crystallization). As implementations became increasingly heterogeneous, so did contributions to the logical design. While it is important to distinguish the physical instantiations from the logical design (OSH licensed), they continue to simultaneously influence one another, despite any divergence. This mutual influence has implications for third-party manufacturers of WR hardware and its implementors, but even more important is the way it affects how insights derived from individual modifications further inform future versions of the WR logical design. This emphasizes the need to restate the often-made point that open source is not about free things (digital, hybrid, and material), but rather the freedom to use and modify what is codified under the open source license (Feller & Fitzgerald, 2001). An important observation is the role that the IEEE1588 PTP standard played in maintaining the cohesion in the logical design. The standard ensured a certain compatibility with complementary technologies in diverse industrial implementations, yet it also anchored some stability and predictability in the evolution of the logical design. This helped mitigate the perceived risk of fragmentation of the WR community that could potentially result with excessive crystallization.

Discussion and Implications

This article investigates the question of how the attributes of a hybrid object and its components affect the open source model of development. Limited insight into the complex nature of hybrids (Kyriakou et al., 2017) has obfuscated where it is possible to organize their developmental work with open source methods traditionally based on digital software.

Our work contributes to the field by conceptualizing hybrid attributes in a manner that extends the ontological separation of digital objects and their material bearers (Faulkner & Runde, 2019; Kallinikos et al., 2013; Yoo et al., 2010). By doing so, we argue how embodiment interacts with granularity and modularity to determine a hybrid object's suitability for an open source development model. We further describe how the endogenous attributes of the hybrid (i.e., its modularity, granularity, and embodiment) interact with exogenous demands; that is, the operating conditions for the object's physical instantiation or any systems integration restrictions. This leads to two sets of implications concerning (1) the role of malleability and material embodiment in digital and hybrid conceptualizations, and (2) liquification and crystallization processes in open source development.

About Malleability and Material Embodiment

First, in a context of increased automation, the Internet of Things, or wearable technologies where the boundary between hardware and software is increasingly blurred (Recker et al., 2021; Romasanta et al., 2021), we argue that the construct of malleability is useful for identifying how the development work of different types of hybrid and digital objects can be organized.

Our analysis qualifies characterizations of hybrids as less malleable compared to digital objects (von Briel et al., 2018; Yoo et al., 2010) that predominantly focus on the effect of the object's material embodiment but underestimate the object's structure. Specifically, a great deal of research in open source development suggests that the physical embodiment of the object can render open source development modalities prohibitive (Balka, 2011; Balka et al., 2010; Boisseau et al., 2018; Oberloier & Pearce, 2018; West & Kuk, 2014). However, following the discussion of von Briel et al. (2018), our case analysis demonstrates that it was the combination of material embodiment, modularity, and granularity of both the hardware and software that determined the hybrid's overall malleability and, consequently, the development model. Although material embodiment is not inconsequential, it is less determinative than the modularity and granularity of the object's structure: In the case of WR, the most important factor was the extreme performance requirements of the core technology that made higher levels of granularity and modularity prohibitive in its logical design, dictating a development model that was, in early phases, centrally controlled and concentrated in a few partners. In subsequent phases, the material embodiment was determinative to the degree that different operating conditions warranted modifications to WR components to make them operationally robust in those environments. These modifications were registered in the WR community and many of them impacted future WR logical designs regulated by the WR OSH license. In this respect, we can very much claim that, due to the nature of the material embodiment and the different physical instantiations of WR, numerous contributions were made to the evolution of the WR logical design.

Extensive material embodiment is often associated with the higher economic cost of manufacturing a physical artifact (Balka, 2011; Balka et al., 2010). It is interesting to note that high development and production costs were not detrimental to WR development as an OSH. Rather, its implementation was simply concentrated in scientific and industrial organizations with the needs and resources to pay for it, just like any other commercial product. In this respect, an OSH with high economic costs mirrors the commercial aspect of open source software 2.0 (Fitzgerald, 2006, Niederman et al., 2006). Although it does not preclude its success, it does more narrowly define the nature of the open source community supporting, using, and monetizing it.

About Liquification and Crystallization

Our analysis also suggests that the malleability of hybrids is not a static property of the object but one that changes through its maturation. Where both modularity and granularity have been applied to describe the structure of digital objects at large (Ekbia, 2009; Kallinikos et al., 2013; Kallinikos & Mariátegui, 2011; Manovich, 2001; Yoo, 2010; Yoo et al., 2010), our study offers a dynamic perspective in which malleability changes across the different phases of its evolution. With high malleability, the processes of liquification or crystallization can result, depending on how the hybrid structural attributes interact with exogenous implementation requirements.

While the term "liquification" (Norman, 2001)—or "liquefaction" (Lusch & Nambisan, 2015)—has previously been employed in the service innovation literature to describe a dematerialization process or "the ability to separate information from the physical world" (Øvrelid & Kempton, 2019, p. 3; Barrett et al., 2012; Lusch & Nambisan, 2015), we extend its meaning to the context of hybrid development to refer to how the process of increasing malleability facilitates a plurality of physical instantiations. Unlike the liquification process, which described a WR transformation resulting in WR variants compatible with the licensed WR logical design, crystallization depicted a process that led to WR variants that were incompatible with the licensed WR logical design.

Liquification and crystallization processes draw parallels with software engineering literature, describing the evolution of purely digital objects (software) through forking processes (Gamalielsson & Lundell, 2014; Robles & González-Barahona, 2012; Zhou et al., 2020). In the development of digital objects, *social forks* describe the processes of creating a public copy of a repository with the goal of contributing to the original project in a distributed development (Ren et al., 2019; Zhou et al., 2020), where the results are compatible variants of the original code (Feitelson, 2012). Within the same literature, *hard forks* describe the process of "splitting off a new development branch" (Zhou et al., 2020) to target user segments or functionality not accommodated in the original version. However, even though the concept of forking is analogous to the liquification and crystallization concepts we propose, their underlying causes are different. In software, hard forks are typically motivated by the fact that as the project matures, the original goals of the contributors eventually diverge, and other contributors may want to take the technology in a different direction (Ven & Mannaert, 2008; Viseur, 2012). In contrast, in our case, the exogenous factors dictating where the hybrid needs to be operated and integrated largely explain the liquification and crystallization processes. Hence, where software forking often occurs as a result of a political or technical disagreement endogenous to the developer community (Robles & González-Barahona, 2012), for hybrids, liquification and crystallization naturally occur in response to exogenous factors (e.g., operating conditions, integration context) that influence the design modifications and subsequently inform the evolution of the hybrid's logical design. This difference is not trivial and further qualifies the role of physical embodiment—and its interaction with exogenous factors—in determining the development modalities.

Practical Implications

Big-science research infrastructures have developed some of the world's most sophisticated technologies with the potential for multiple and unanticipated applications in different industries (Pujol Priego et al., 2022; Wareham & Pujol Priego, 2019; Wareham et al., 2022). In parallel, commercial interest in OSH is growing, particularly for organizations that want to minimize the nonrecurring engineering costs of technologies that do not yet exist by sharing these expenses with an open source community (Barrett et al., 2013; Deodhar et al., 2012; Fitzgerald, 2006; Spaeth et al., 2014; von Krogh et al., 2012). An important opportunity exists to develop such sophisticated technologies through open source development, unlocking its potential for downstream product developments. While skepticism prevails in policy and investment circles regarding how highly sophisticated technology stacks may be developed following open source premises, our analysis suggests that development models can adjust to the changing malleability of the object through time. The WR case acknowledges a need, at specific points, for traditional open source development to incorporate more centralized, directed, and less bazaar-like processes.

The adjustment of the open source premise with more directed development has already been identified in more commercially oriented open source software (Dahlander & O'Mahony, 2010; Fitzgerald, 2006), where commitments and clear development decisions sometimes need to prevail over heterogeneous viewpoints and self-sectioning (Bergquist & Ljungberg, 2001; Casadesus-Masanell & Ghemawat, 2006; Raymond, 1999). Our work qualifies these insights for commercially focused OSH, giving physical embodiment a role that is determinative yet subordinate. If we liberate open source from the prejudices that have traditionally shadowed hardware (i.e., high economic costs of production), we can better cultivate the knowledge and practice of how sophisticated, commercially adopted hardware can be developed following open source premises.

Limitations and Future Research

The selection of a single case such as WR limits the generalizability of the findings. Nonetheless, (1) we employed different data collection methods and collected a sufficient amount of secondary and primary data from multiple informants to increase confidence in our interpretations, (2) we took care in documenting our analytical processes to dissect our inferences from the empirical data, (3) we incorporated diverse viewpoints and interpretations, and (4) we shared and discussed our findings with the WR community at two different stages of our analytical progression to corroborate our interpretations. Such processes helped build confidence in our analytical steps. Although this single case study design allowed us to generalize to a theory (Lee & Baskerville, 2003; Tsang & Williams, 2012), we acknowledge that further research is required to replicate our findings across different contexts (Yin, 2003).

First, we should be prudent in extrapolating our findings to contexts that do not have the same level of technical sophistication, economic resources, and political status as the sponsor CERN, as these factors may be influential in the case of WR. Second, as a technology, there are two aspects of WR that are also exceptional. Because time measurement in the extreme is very sensitive to both the physical and logical design of the technology, other OSH projects may not have the same technical sensitivities and may therefore be amenable to a wider range of development models. Additionally, WR is not a general-use technology like an operating system or scripting software; it was commissioned with a specific purpose and is intolerant to significant variance in its performance. Clearly, OSH projects that are more general purpose and unconstrained by such rigid performance requirements may tolerate greater scope drift or more organic development processes. We hope our study will trigger more empirical OSH research and will open new opportunities for future research on hybrid malleability. In our study, we were able to observe the trajectory (i.e., the increasing or decreasing) of malleability, but theoretical and empirical work that takes the next step and investigates malleability patterns (e.g., linearity, exponentiality) can help us better understand the implications of how malleability behaves for development practices.

Conclusion

Open source hardware is growing in its visibility and promise. However, limited experience and insight into OSH for larger commercial applications has hindered the realization of its full potential. Seeking a more nuanced and useful understanding, our research focuses on a holistic continuum of physical and digital attributes, allowing us to better understand the determinants of open source development for hybrid technology objects. Our work challenges the assumptions and addresses some of the shortcomings of prior research in OSH, particularly the focus on physical embodiment as detrimental to open source development. Alternatively, we offer a more complex explanation of how an object's physical embodiment interacts with other structural attributes that determine its amenability to open source development. Our theoretical insights can potentially trigger new ways of looking at both digital and hybrid objects and offer new avenues to leverage the potential of OSH.

Acknowledgments

We are deeply indebted to the senior editor, Sirkka Jarvenpaa, and the associate editor, Jan Recker, for their editorial guidance and developmental feedback, which helped us shape our work and improve the paper. We are also indebted to the team of anonymous reviewers for their insightful comments and suggestions. We also appreciate the support of all the people we had the privilege to meet at CERN, as well as the members of the White Rabbit community, who were a fundamental piece of the study. Preliminary findings from this study were presented at the 39th International Conference on Information Systems (ICIS 2018).

References

Ackerman, J. R. (2008). Toward open source hardware. *University of Dayton Law Review, 34,* 183-222.

Ågerfalk, P. J., & Fitzgerald, B. (2008). Outsourcing to an unknown workforce: Exploring open sourcing as a global sourcing strategy. *MIS Quarterly, 32*(2), 385-409. https://doi.org/10.2307/25148845

Akhlaghpour, S., Wu, J., Lapointe, L., & Pinsonneault, A. (2013). The ongoing quest for the IT artifact: Looking back, moving forward. *Journal of Information Technology, 28*(2), 150-166. https://doi.org/10.1057/jit.2013.10

Allen, R. C. (1983). Collective invention. *Journal of Economic Behavior and Organization, 4*(1), 1-24. https://doi.org/10.1016/0167-2681(83)90023-9

Balka, K. (2011). *Open source product development: The meaning and relevance of openness.* Springer.

Balka, K., Raasch, C., & Herstatt, C. (2010). How open is open source? Software and beyond. *Creativity and Innovation Management, 19*(3), 248-256. https://doi.org/10.1111/j.1467-8691.2010.00569.x

Barrett, M., Davidson, E., Fayard, A.-L., Vargo, S., & Yoo, Y. (2012, December 14). Being innovative about service innovation: Service, design and digitalization. In *Proceedings of the International Conference on Information Systems*. https://aisel.aisnet.org/icis2012/proceedings/Panels/7

Barrett, M., Heracleous, L., & Walsham, G. (2013). A rhetorical approach to IT diffusion: Reconceptualizing the ideology-framing relationship in computerization movements. *MIS Quarterly, 37*(1) 201-220. http://www.jstor.org/stable/43825943

Benkler, Y. (2002). Coase's penguin, or, Linux and "the nature of the firm." *Yale Law Journal, 112*(3) Article 369. https://doi.org/10.2307/1562247

Benkler, Y. (2006). *The wealth of networks: How social production transforms markets and freedom.* Yale University Press.

Bergquist, M., & Ljungberg, J. (2001). The power of gifts: Organizing social relationships in open source communities. *Information Systems Journal, 11*(4), 305-320. https://doi.org/10.1046/j.1365-2575.2001.00111.x

Bogers, M., & Horst, W. (2014). Collaborative prototyping: Cross-fertilization of knowledge in prototype-driven problem solving. *Journal of Product Innovation Management, 31*(4), 744-764. https://doi.org/10.1111/jpim.12121

Boisseau, É., Omhover, J.-F., & Bouchard, C. (2018). Open-design: A state of the art review. *Design Science, 4*(e3). https://doi.org/10.1017/dsj.2017.25

Bonvoisin, J., Mies, R., Boujut, J.-F., & Stark, R. (2017). What is the "source" of open source hardware? *Journal of Open Hardware, 1*(1), Article 5. https://doi.org/10.5334/joh.7

Boujut, J.-F., Pourroy, F., Marin, P., Dai, J., & Richardot, G. (2019). Open source hardware communities: investigating participation in design activities. In *Proceedings of the Design Society: International Conference on Engineering Design, 1*(1), 2307-2316.

Bryant, A., & Charmaz, K. (2007). *The SAGE Handbook of Grounded Theory*. SAGE. https://dx.doi.org/10.4135/9781848607941

Casadesus-Masanell, R., & Ghemawat, P. (2006). Dynamic mixed duopoly: A model motivated by Linux vs. Windows. *Management Science, 52*(7), 1072-1084. https://doi.org/10.1287/mnsc.1060.0548

Cook, J. (2001). *Open source development: An Arthurian legend*. Presented at the 1st Workshop on Open-source Software Engineering at ICSE 2001, Toronto, ON, Canada. https://flosshub.org/content/open-source-development-arthurian-legend

Crowston, K. (1997). A coordination theory approach to organizational process design. *Organization Science, 8*(2), 157-175. https://doi.org/10.1287/orsc.8.2.157

Crowston, K., & Howison, J. (2006). Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology and Policy, 18*(4), 65-85. https://doi.org/10.1007/s12130-006-1004-8

Cusumano, M. A. (1992). Shifting economies: From craft production to flexible systems and software factories. *Research Policy, 21*(5), 453-480. https://doi.org/10.1016/0048-7333(92)90005-O

Dahlander, L., & O'Mahony, S. (2010). Progressing to the center: Coordinating project work. *Organization Science, 22*(4), 961-979. https://doi.org/10.1287/orsc.1100.0571

DeMicheli, G., & Sami, M. G. (2013). *Hardware/software co-design.* Springer.

Deodhar, S. J., Saxena, K. B. C., Gupta, R. K., & Ruohonen, M. (2012). Strategies for software-based hybrid business models. *The Journal of Strategic Information Systems, 21*(4), 274-294. https://doi.org/10.1016/j.jsis.2012.06.001

Dourish, P. (2001). *Where the action is: The foundations of embodied interaction.* MIT Press. https://doi.org/10.7551/mitpress/7221.001.0001

Drechsler, R., & Breiter, A. (2007, July 22-25). Hardware project management: What we can learn from the software development process for hardware design? In *Proceedings of the 2nd International Conference on Software and Data Technologies* (pp. 409-416). https://doi.org/0.5220/0001324204090416

Ekbia, H. R. (2009). Digital artifacts as quasi-objects: Qualification, mediation, and materiality. *Journal of the American Society for Information Science and Technology, 60*(12), 2554-2566. https://doi.org/10.1002/asi.21189

Faulkner, P., & Runde, J. (2009). On the identity of technological objects and user innovations in function. *Academy of Management Review, 34*(3), 442-462. https://doi.org/10.5465/amr.2009.40632318

Faulkner, P., & Runde, J. (2013). Technological objects, social positions, and the transformational model of social activity. *MIS Quarterly, 37*(3), 803-818. https://www.jstor.org/stable/43826001

Faulkner, P., & Runde, J. (2019). Theorizing the digital object. *MIS Quarterly, 43*(4), 1279-1302. 10.25300/MISQ/2019/13136

Feller, J., & Fitzgerald, B. (2000). A framework analysis of the open source software development paradigm. In *Proceedings of the 21st International Conference on Information Systems* (pp. 58-69).

Feller, J., & Fitzgerald, B. (2001). *Understanding open source software development* (1st ed.). Addison-Wesley Professional.

Feitelson, D. (2012). Perpetual development: A model of the Linux kernel life cycle. *Journal of Systems and Software, 85*(4), 859-875. https://doi.org/10.1016/j.jss.2011.10.050

Fitzgerald. (2006). The transformation of open source software, *MIS Quarterly, 30*(3), 587. https://www.jstor.org/stable/25148740

Gajski, D. D., & Vahid, F. (1995). Specification and design of embedded hardware-software systems. *IEEE Design Test of Computers, 12*(1), 53-67. https://doi.org/10.1109/54.350695

Gamalielsson G., & Lundell, B. (2014). Sustainability of open source software communities beyond a fork: How and why has the LibreOffice project evolved? *Journal of Systems and Software, 89,* 128-145. https://doi.org/10.1016/j.jss.2013.11.1077

Gerring, J. (2007). *Case study research: Principles and practices.* Cambridge University Press.

Gibb, A. (2014). *Building open source hardware: DIY manufacturing for hackers and makers* (1st ed.). Addison-Wesley Professional.

Gioia, D. A., Corley, K. G., & Hamilton, A. L. (2013). Seeking qualitative rigor in inductive research: Notes on the Gioia methodology. *Organizational Research Methods*, *16,* 15-31. https://doi.org/10.1177/1094428112452151

Howison, J., & Crowston, K. (2014). Collaboration through open superposition: A theory of the open source way. *MIS Quarterly, 38*(1), 29-50. https://www.jstor.org/stable/26554867

Kallinikos, J., Aaltonen, A., & Marton, A. (2010). A theory of digital objects. *First Monday, 15*(6). https://doi.org/10.5210/fm.v15i6.3033

Kallinikos, J., Aaltonen, A., & Marton, A. (2013). The ambivalent ontology of digital artifacts. *MIS Quarterly, 37*(2), 357-370. https://doi.org/10.25300/MISQ/2013/37.2.02

Kallinikos, J., & Mariátegui, J.-C. (2011). Video as digital object: Production and distribution of video content in the internet media ecosystem. *The Information Society, 27*(5), 281-294. https://doi.org/10.1080/01972243.2011.607025

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly 23*(1), 67-93. https://doi.org/10.2307/249410

Kyriakou, H., Nickerson, J.V. & Sabnis, G., (2017). Knowledge reuse for customization: Metamodels in an open design community for 3D printing. *MIS Quarterly*, *41*(1), 315-332. https://doi.org/10.25300/MISQ/2017/41.1.17

Lakhani, K. R., & von Hippel, E. (2004). How open source software works: "Free" user-to-user assistance. In C. Herstatt & J. G. Sander (Eds.), *Produktentwicklung mit virtuellen Communities: Kundenwünsche erfahren und Innovationen realisieren* (pp. 303-339). Gabler. https://doi.org/10.1007/978-3-322-84540-5_13

Lee, A. S., & Baskerville, R. L. (2003). Generalizing generalizability in information systems research. *Information systems research*, *14*(3), 221-243. https://doi.org/10.1287/isre.14.3.221.16560

Leonardi, P. M. (2010). Digital materiality? How artifacts without matter, matter. *First Monday*, *15*(6-7). https://doi.org/10.5210/fm.v15i6.3036

Lindberg, A., Berente, N., Gaskin, J., & Lyytinen, K. (2016). Coordinating interdependencies in online communities: A study of an open source software project. *Information Systems Research, 27*(4), 751-772. https://doi.org/10.1287/isre.2016.0673

Lipiński, M., Włostowski, T., Serrano, J., & Alvarez, P. (2011, September 12-16). White Rabbit: A PTP application for robust sub-nanosecond synchronization. In *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication* (pp. 25-30). https://ieeexplore.ieee.org/document/6070148

Lok, J., & de Rond, M. (2012). On the plasticity of institutions: Containing and restoring practice breakdowns at the Cambridge University Boat Club. *Academy of Management Journal, 56*(1), 185-207. https://doi.org/10.5465/amj.2010.0688

Lusch, R. F., & Nambisan, S. (2015). Service innovation: A service-dominant logic perspective. *MIS Quarterly, 39*(1), 155-176. https://doi.org/10.25300/MISQ/2015/39.1.07

Manovich, L. (2001). *The language of new media.* MIT Press.

Markus, M. L. (2007). The governance of free/open source software projects: Monolithic, multidimensional, or configurational? *Journal of Management and Governance, 11*(2), 151-163. https://doi.org/10.1007/s10997-007-9021-x

Masum, H. (2000). Reputation layers for open-source development. In *Making sense of the bazaar: Proceedings of the 1st workshop open source software engineering* (pp. 3-5). https://flosshub.org/sites/flosshub.org/files/masum.pdf

Mellis, D., & Buechley, L. (2012). Collaboration in open-source hardware: Third-party variations on the Arduino Duemilanove. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (pp. 1175-1178). https://doi.org/10.1145/2145204.2145377

Mies, R., Bonvoisin, J., & Jochem, R. (2019). Harnessing the synergy potential of open source hardware communities. In T. Redlich, M. Moritz, & J. Wulfsberg (Eds.), *Co-creation: Reshaping business and society in the era of bottom-up economics* (pp. 129-145). Springer. https://doi.org/ 10.1007/978-3-319-97788-1_11

Miles, M. B., & Huberman, A. M. (1994). Qualitative data analysis: An expanded sourcebook (2nd ed.). SAGE.

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309-346. https://doi.org/10.1145/567793.567795

Moreira, P., Serrano, J., Wlostowski, T., Loschmidt, P., & Gaderer, G. (2009, October 12-16). White Rabbit: Sub-nanosecond timing distribution over ethernet. In *Proceedings of the ISPCS 2009 International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication,* 58-62. https://doi.org/10.1109/ISPCS.2009.5340196

Niederman, F., Davis, A., Greiner, M. E., Wynn, D., & York, P. T. (2006). Research agenda for studying open source II: View through the lens of referent discipline theories. *Communications of the Association for Information Systems, 18*(7) 129-149. https://doi.org/10.17705/1CAIS.01808

Norman, R. (2001). *Reframing business: When the map changes the landscape.* Wiley.

Oberloier, S., & Pearce, J. M. (2018). General design procedure for free and open-source hardware for scientific equipment. *Designs, 2*(1), 2. https://doi.org/10.3390/designs2010002

O'Mahony, S., & Ferraro, F. (2007). The emergence of governance in an open source community, *Academy of Management Journal. 50*(5), 1079-1106. https://doi.org/10.5465/amj.2007.27169153

Open Source Hardware Association. (2012). *Definition of open source hardware.* Open Source Hardware Association. https://www.oshwa.org/definition/

Orlikowski, W. J., & Iacono, C. S. (2001). Research commentary: Desperately seeking the "IT" in IT Research—A call to theorizing the IT artifact. *Information Systems Research, 12*(2), 121-134. https://doi.org/10.1287/isre.12.2.121.9700

Øvrelid, E., & Kempton, A. (2019). From recombination to reconfiguration: Affording process innovation in digital infrastructures. In *Proceedings of the 27th European Conference on Information Systems.*

Paavola, S., & Miettinen, R. (2019). Dynamics of design collaboration: BIM models as intermediary digital objects. *Computer Supported Cooperative Work, 28*(1), 1-23. https://doi.org/10.1007/s10606-018-9306-4

Pan, W., Li, Z., Zhang, Y., & Weng, C. (2018). The new hardware development trend and the challenges in data management and analysis. *Data Science and Engineering, 3*(3), 263-276. https://doi.org/10.1007/s41019-018-0072-6

Pearce, J. M. (2012). Building research equipment with free, open-source hardware. *Science, 337*(6100), 1303-1304. https://doi.org/10.1126/science.1228183

Pearce, J. M. (2017). Emerging business models for open source hardware. *Journal of Open Hardware, 1*(1). https://doi.org/10.5334/joh.4

Pujol Priego, L., Wareham, J., & Romasanta, A.K. (2022) The puzzle of sharing scientific data, *Industry and Innovation*, *29*(2), 219-250. https://doi.org/10.1080/13662716.2022.2033178

Raymond, E. (1999). The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary. *Knowledge, Technology and Policy, 12*(3), 23-49.

Recker, J., Lukyanenko, R., Jabbari, M., Samuel, B. M., & Castellanos, A. (2021). From representation to mediation: A new agenda for conceptual modeling research in a digital world. *MIS Quarterly*, *45*(1), 269-300. https://doi.org/10.25300/MISQ/2021/16027

Ren, L., Zhou, S., Kästner, C. & Wą sowski, A. (2019). Identifying redundancies in fork-based development. In *Proceedings of the IEEE 26th International Conference on Software Analysis, Evolution, and Reengineering* (pp. 230-241). https://doi.org/10.1109/SANER.2019.8668023

Robles, G., González-Barahona, J.M. (2012). A comprehensive study of software forks: Dates, reasons and outcomes. In I. Hammouda, B. Lundell, T. Mikkonen, & W. Scacchi (Eds.), *Open source systems: Long-term sustainability.* Springer. https://doi.org/10.1007/978-3-642-33442-9_1

Romasanta, A., Ahmadova, G., Wareham, J. & Pujol Priego, L. (2021). Deep tech: Unveiling the foundations (ESADE working papers series 276). Available at https://ssrn.com/abstract=4009164

Sanchez, R., & Mahoney, J. T. (1996). Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management Journal, 17*(S2), 63-76. https://doi.org/10.1002/smj.4250171107

Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S., & Lakhani, K. (2006). Understanding free/open source software development processes. *Software Process: Improvement and Practice, 11*(2), 95-105. https://doi.org/10.1002/spip.255

Shah, S. K. (2005). Open beyond software. In C. DiBona, D. Cooper, & M. Stone (Eds.), *Open sources 2.0: The continuing evolution* (pp. 339-360). O'Reilly Media.

Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science, 52*(7), 1000-1014. https://doi.org/10.1287/mnsc.1060.0553

Shaikh, M., & Vaast, E. (2016). Folding and unfolding: Balancing openness and transparency in open source communities. *Information Systems Research, 27*(4), 813-833. https://doi.org/10.1287/isre.2016.0646

Spaeth, S., von Krogh, G., & He, F. (2014). Research note: Perceived firm attributes and intrinsic motivation in sponsored open source software projects. *Information Systems Research, 26*(1), 224-237. https://doi.org/10.1287/isre.2014.0539

Tsang, E. W., & Williams, J.N., (2012). Generalization and induction: Misconceptions, clarifications, and a classification of induction. *MIS Quarterly*, 729-748. https://doi.org/10.2307/41703478

Ven, K., & Mannaert, H. (2008). Challenges and strategies in the use of open source software by independent software vendors. *Information and Software Technology, 50*(9-10), 991-1002. https://doi.org/10.1016/j.infsof.2007.09.001

Viseur, R. (2012). Forks impacts and motivations in free and open source projects. *International Journal of Advanced Computer Science and Applications, 3*(2), 117-122. https://doi.org/10.14569/IJACSA.2012.030221

von Briel, F., Recker, J., & Davidsson, P. (2018). Not all digital venture ideas are created equal: Implications for venture creation processes. *The Journal of Strategic Information Systems, 27*(4), 278-295. https://doi.org/10.1016/j.jsis.2018.06.002

von Hippel, E. & von Krogh, G. (2003). Open source software and the "private-collective" innovation model: Issues for organization science. *Organization Science, 14*(2), 209-223. https://doi.org/10.1287/orsc.14.2.209.14992

von Krogh, G. Haefliger, S., Spaeth, S., & Wallin, M. W. (2012). Carrots and rainbows: Motivation and social practice in open source software development. *MIS Quarterly, 36*(2), 649. https://doi.org/10.2307/41703471

Wareham, J., & Pujol Priego, L. (2019). From big science to big business. *Research Professional News*. https://www.researchprofessionalnews.com/rr-news-europe-views-of-europe-2019-6-from-big-science-to-big-business/

Wareham, J., Pujol Priego, L., Romasanta, A.K., Mathiassen, T.W., Nordberg, M., & Tello, P.G., (2022). Systematizing serendipity for big science infrastructures: The ATTRACT project. *Technovation,* 102374. https://doi.org/10.1016/j.technovation.2021.102374

West, J., & Kuk, G. (2014). Proprietary benefits from open communities: How MakerBot leveraged Thingiverse in 3D printing. *3D Printing, 27*. Available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2544970

Yin, R. K. (2003). *Case study research: Design and methods (applied social research methods).* SAGE.

Yoo, Y. (2010). Computing in everyday life: A call for research on experiential computing. *MIS Quarterly, 34*(2), 213-231. https://doi.org/10.2307/20721425

Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). Research commentary—The new organizing logic of digital innovation: An agenda for information systems research. *Information Systems Research, 21*(4), 724-735. https://doi.org/10.1287/isre.1100.0322

Yu, F., Pasinelli, M., & Brem, A. (2018). Prototyping in theory and in practice: A study of the similarities and differences between engineers and designers. *Creativity and Innovation Management, 27*(2), 121-132. https://doi.org/10.1111/caim.12242

Zhou, S., Vasilescu, B. & Kästner, C., (2020). How has forking changed in the last 20 years? A study of hard forks on GitHub. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (pp. 445-456). https://cmustrudel.github.io/papers/zhou20forks.pdf

About the Authors

Laia Pujol Priego is an assistant professor at ESADE Business & Law Schools, Ramon Llull University. She is also a research fellow at The Lisbon Council, a Brussels-based think tank. Laia's research program focuses on digital innovation, in particular, the management implications of developing and commercializing emerging and science-based technologies, data commons infrastructures and data practices. Her research has appeared in *Technovation, Government Information Quarterly, Industry and Innovation, Issues of Science and Technology,* and *Innovation: Organization & Management.* Laia's past affiliations include IESE Business School, International University of Catalonia (UIC), and was visiting researcher at McDonough Business School at Georgetown University. Her research has been financially supported by the European Commission, the Catalan government, the University of Deusto, Comillas Pontifical University, and Ramon Llull University.

Jonathan Wareham is a professor/catedrático of information systems at ESADE Business & Law Schools, Ramon Llull University, where he previously served as dean (Faculty & Research) and vice dean (Research). His current research focuses on scientific computing, scientific policy, and how scientific technologies are influencing the business world. Wareham's research has been published in journals and proceedings such as *Organization Science, Decision Sciences, MIS Quarterly, Decision Support Systems, IEEE Transactions on Engineering Management, IEEE Computer, Communications of the ACM, Journal of the American Society for Information Science and Technology*, and others. He has served as a senior editor for *MIS Quarterly* and as an associate editor for *Information Systems Research.* He was the general conference chair of the 20th European Conference on Information Systems (ECIS 2012). Personal website: https://jwareham.org/

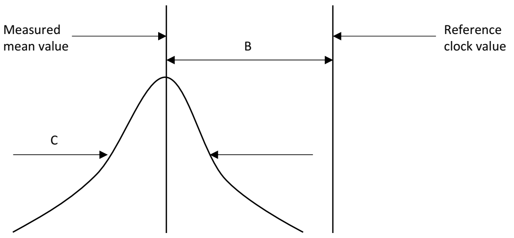
Appendix A

Technical Note

Table A1. Technical Description of WR Components

Switch components	**Description**	**Type**
WR switch box	WR switch box is a white metal 19' 1 U case with two cooling fans in the back.	Hardware
Main PCB: Contains the main electronics components, ARM processor, Xilinx FPGA chip, oscillators, memories, etc.	PCB is a printed circuit board and FPGA is a field programmable gate array. ARM processor is a central processing unit (CPU) built on the RISC-based architecture developed by Advanced RISC Machines (thus ARM). Oscillators are devices for generating oscillatory electric currents or voltages by non-mechanical means. Memories refer to devices that are used to store information for immediate use.	
Backplane PCB: Contains electrical connections to 18 SFP cages, debug USB-UART ports, Light-emitting diode (LEDs), etc.	Large-format printed circuit boards (PCBs) are used as backbones for connecting several PCBs together. USB = universal serial bus, and UART corresponds to universal asynchronous receiver/transmitter. USB-UART ports are controllers that provide USB connectivity to devices with a UART interface.	
General-purpose gateware: IP cores used both in the switch	IP cores are intellectual property core or preconfigured logic functions implemented in the switch.	Gateware
Dedicated switch gateware	Package contains field-programmable gate array gateware running in the embedded Linux.	
Gateware-software interface	Gateware-software interfaces contain wishbone bus configuration registers of the modules inside the gateware of the WR switch.	
Dedicated switch software	Package contains field-programmable gate array software running in the embedded Linux.	Software
at91bootstrap-3.3	Second-level bootloader for Microchip SoC (system on a chip) provides a set of algorithms to manage the hardware initialization such as clock speed configuration.	
barebox-2014.04	Primary boot loader is used in embedded devices.	
Linux-3.16.38	Linux kernel package.	
buildroot-2016.02	Make files and patches that facilitate the generation of a complete and insignificant embedded Linux system.	
General-purpose software: Used both in the switch and node	Precise Time Protocol (PTP) is a software stack whose single source code can be compiled for many architectures and which is easily extensible.	
Node components	***Description***	
WR PTP core and SFP	PTP is the Precision Time Protocol implemented on a FPGA mezzanine card. SFP is a small form-factor pluggable: a compact, hot-pluggable network interface module.	Hardware
General-purpose gateware: IP cores used both in the node	Intellectual property cores or preconfigured logic functions are used in the node.	Gateware
Dedicated gateware for the node	Specific gateware layer allows WR to be used as a standard network interface card implementing the WR technology functionalities.	
General-purpose software: Used both in the switch and node	Precise Time Protocol (PTP) is a software stack whose single source code can be compiled for many node architectures.	Software
Dedicated software for the node	Specific software layer allows WR to be used as a standard network interface card implementing the WR technology functionalities.	

Table A2. Technical Definitions

Technical term	Full description
Accuracy	The mean over an ensemble of measurements of the time or frequency error between the clock under test and a reference clock. Line B in Figure below represents the error in the measured mean value with respect to the reference, i.e., the accuracy. The width of the curve of ensemble measurements is represented by line C. 
Boundary clock	A PTP instance that has multiple PTP ports in a domain and maintains the timescale used in the domain. Within a domain, it may serve as the source of time to other PTP instances, i.e., be a master clock, and can in addition synchronize to another boundary clock or ordinary clock, i.e., be a slave clock.
Clock	A device that can provide a measurement of the passage of time since a defined epoch. A clock provides time at desired moments of the timescale it maintains. Time is obtained either: Physically: In this type of clock, time is modeled using a clock signal and a time counter that is driven by the clock signal. Mathematically: In this type of clock, time is generated by a model that describes the relation of this clock to another clock (e.g., to a physical clock in a different timescale). The model enables the calculation of the time of the clock from the time of the other clock.
Clock signal	A physical signal that has periodic events. The periodic events mark the significant instants at which a time counter is incremented. The clock signal is characterized by its frequency and phase.
Epoch	The origin of a timescale.
Event	An abstraction of the mechanism by which signals or conditions are generated and represented. In this abstraction, the aspects of interest of the signals are conditions that occur at discrete instants of time.
Grandmaster clock	In the context of a single PTP domain, the local PTP clock of an ordinary clock or a boundary clock that is the source of time to which all other local PTP clocks in the domain are synchronized.
Ordinary clock	A PTP instance that has a single PTP port in its domain and maintains the 208 timescales used in the domain. An ordinary clock can serve as a source of time, i.e., contain a master clock; or, alternatively, the local PTP clock of an ordinary clock can be synchronized, i.e., be a slave clock to the local PTP clock of a boundary clock or another ordinary clock in the domain.
Precision	Precision is the degree to which repeated (or reproducible) measurements under unchanged conditions show the same results.
Synchronized clocks	Absent relativistic effects, two clocks are synchronized to a specified uncertainty if they have the same epoch and their measurements of the time of the same single event occurring at an arbitrary instant differ by no more than that uncertainty.

Note: Technical definitions were extracted from the WR standard draft: P1588/D1.3, *Draft Standard for a Precision Clock Synchronisation Protocol for Networked Measurement and Control Systems* (June 2018), Version 3.08.

Appendix B

Method Note

Examples of interview guide through an advanced stage of the analytical progression, corresponding to analytical Phase 3 in Table A2. Summary of primary data collected.

Table B. Example of Interview Guide Through Advance Stage of the Analytica Progression:
Initial engagement
How did you learn about and initially become involved in WR?
When did you engage in WR development?
What has been your role and the role of your organization in WR development? Which components have you and your organization contributed to?
How did your organization fund the investment for collaborating in WR? Did it change over time? (If it was via a contract): What was the reason for the contract? Duration? What happened after the contract?
About WR technology
What are the components, functions, and applications of WR?
How would you describe the WR switch? What is the structure of the switch?
How would you describe the WR node? What is the structure of the node?
How does the structure of the switch and node differ?
Did the structure of the switch and node change? If yes, why?
What were the main differences in the switch versions that emerged?
What were the main differences in the node versions that emerged?
How do you explain the number of versions in both the switch and node?
About the process of development of the different components
Could you describe the main steps in the development of WR that you recall?
What, in your opinion, were the major events in the development of WR? Why?
In which phases were you involved in WR development?
If respondent was involved in the specifications: How did you agree on WR specifications?
If respondent was involved in the design of the switch: How was the design of WR switch organized?
If respondent was involved in the design of the node: How was the design of WR node organized?
How was the production of WR prototypes was managed?
If respondent was involved in the testing: How was WR testing and certification organized?
If respondent was involved in the standardization activities: How was WR standardization organized?
Did you report to anyone inside and outside your organization?
How did you coordinate your work with other contributors in your organization?
Did you select your own tasks? Which tasks?
Which tools did you use to develop and communicate the outcomes of your work? How did you use the WR repository, WR Wiki, and mailing list? Others?
Did you have WR meetings? With whom? For what purpose?
Did you develop any proprietary WR? If yes: When? Why? How did you organize the proprietary development?

Table B2. Summary of Interview Data Collected

Stage	Respondent	Interviews	Role
Phase 1: General understanding of WR structure, the process, different phases, agents, and actions in WR development	RSE1	2	Engineer in research infrastructures (RI)
	RSE2	2	Engineer in RI
	RSE3	1	Engineer in RI
	RT1	1	Personnel at the technology transfer offices (TTO)
	RT2	1	Personnel at the TTO
	P1	1	Personnel at the TTO
	R1	1	Scientist/engineer in RI
	R2	1	Scientist/engineer in RI
	R3	1	Scientist/engineer in RI
	CS1	1	Engineer in company developing SW
	CH1	1	Engineer in company developing HW
	CH2	1	Engineer in company developing HW
	CH3	1	Engineer in company developing HW
	CH4	1	Engineer in company developing HW
	CD1	1	Engineer in company developing WR pilots
	CA1	1	Customers of WR not involved in WR development
Phase 2: Understanding how work was organized in the WR development process within the different phases identified	RSE4	2	Engineer at RI
	RSE 5	1	Engineer at RI
	RSE1	1	Engineer at RI
	R3	1	Scientist/engineer in RI
	R4	1	Scientist/engineer in RI
	R5	1	Other staff in RI involved in WR
	CS2	1	Engineer in RI
	CH4	1	Engineer in company developing HW
	CD3	1	Engineer in company developing WR pilots
	R6	1	Engineer in RI
	CA1	1	Customers of WR not involved in WR development
Phase 3: Increasing detail on the characterization of WR structure and each development model	RSE1	1	Scientist/engineer in RI
	RSE 5	1	Scientist/engineer in RI
	RSE 4	1	Scientist/engineer in RI
	CH1	1	Engineer in company developing HW
Phase 4: Confirmation of the interpretations and fine-grained detail on WR attributes and relationships with the development characteristics	CH5	1	Engineer in company developing HW
	CH6	1	Engineer in company developing HW
	CH7	1	Engineer in company developing HW
	RSE 4	1	Engineer in RI

Table B3. Illustration of Empirical Analysis of WR Development

Aggregate dimension: Malleability	
Second-order codes	***Selected evidence on first-order codes from representative quotes and empirical observations***
Embodiment	"We developed (in collaboration with CERN, GSI, and other partners) the 18-port White Rabbit Switch, designing the main board in the MicroTCA form factor. The core element was a Virtex-6 (LX240T) FPGA. We paired this device with an external processor (ARM926E) running an embedded Linux OS to perform the high-level operations such as system updates, file management, etc. The switch uses 18 GTX links for SFPs and 40 GPIOs for general-purpose tasks (LEDS, SFP detection, etc.)."[14]
	"If we think of a node, we think about an IP core that you can instantiate in different hardware." RSE2
	"If we say a switch, we think about a hardware box." RSE1
Granularity	"You could not make it (the switch) more granular; it would make it many costs and extra work and less efficacy in terms of precision. It would be harder to make it work."RSE1
	"Gateware and software are easier to split it among companies, but hardware does not make sense." RSE6
Modularity	"Node is an end device whether it receives or sends staff to one port. You throw or you digest the data. It is like one of the switch ports; plus, you need to implement, like in the switch, WR protocol" RSE 6
	"The switch is quite a compact device; it needs to work like one unified device, and if you have different companies, you need to define different interfaces between the parts that they are designing, test each part, see that they work together, and you make it much more complex, too much work and much more expensive. For the precision, it is also better that you do not have so many connectors; here, it was not practical." RSE 1
Aggregate dimension: Liquification	
Second-order codes	***Selected evidence on first-order codes from representative quotes and empirical observations***
Logical design	"It needs to allow different configurations…plus, you need to implement more flexibility because it needs to allow different types of configurations; plus, you need to implement more features let's say." "The exact configuration depends on application requirements." RSE6 "I said I'd open a thread here to ask for suggestions regarding the feature list for a new version of the WR switch. Considering it usually takes a couple of years to go from ideas to a product you can actually buy, I think it's good to launch an informal discussion about the features early enough."[15]
Development	"We had different actors working in parallel. I was coordinating the contributions that came from gateware Y that was integrating everything together. In the beginning, we had two companies helping with the software and gateware, the other two for the hardware. X was integrating everything."RSE6 "We are now developing gateware for new designs of the nodes, so we are supporting different applications of the nodes because it depends on each application." RSE6 "The node is different because the first node was a spec board and designed here, and then one company developed a simplified version. Some people took this design and made different formats, and this was without us doing it, we did not pay for the design, it was because people needed it." RSE6 "In embedded detector electronics there is usually not much space available. A standard WR switch is not suited for detectors like Chromium or KM3NeT." (WR repository, 2020) "There was an engineer from South Africa who was interested in solving particular problems with much greater temperature variations in South Africa compared to CERN, where optical fibres are underground and naturally isolated from the temperature variations of the atmosphere. So, he discovered effects that we had not seen and voluntarily improved WR in what affects timing for long distances." RSE1

[14] In Xcell Journal issue 91: https://issuu.com/xcelljournal/docs/xcell_journal_issue_91
[15] In WR repository: https://www.ohwr.org/project/wr-switch-hw-v4/wikis/uploads/f677af5cb169e3b031c33cf5ed768ac8/msg00015.html

Physical instantiations	"CRIO-WR is a standalone White Rabbit node implementation on a PCB with a form factor for National Instruments CompactRIO modules. The board is originally derived from and keeps maximum firmware compatibly with the established boards SPEC and CUTE-WR." (WR repository, 2021) • LHAASO—New node design (Tsinghua University, China) implements WR • CTA—Cherenkov Telescope Array, implementation of WR • The first deployment of a system based on WR synchronization in Gran Sasso (CNGS) measurements • China Spallation Neutron Source Institute of High Energy Physics, CSNS implements WR • CNGS. Timing for neutrino measurements implements WR • DESY, Germany implements WR • MIKES (Center for Metrology and Accreditation, Finland) implements WR. Switch and node design improvements • Dept of Physics and Astronomics VUA, The Netherlands
Aggregate dimension: Crystallization	
Second-order codes	***Selected evidence on first-order codes from representative quotes and empirical observations***
Logical design	"It is easier, less effort and cheaper to adapt the design of a company by adding some components and you would have your design with WR support. It would be extensive and massive work to add our open source implementation. This would imply that all user interfaces and many features that we don't have in open source. It would imply a lot of development and testing" RSE6 "If you have your own implementation of PTP [it] is less work to add/modify some components than change everything to do it using the open source components. It would be extensive and massive work to add our open source implementation. This would imply that all user interfaces and many features that we don't have in open source. It would imply a lot of development and testing." CH5 "Also, as there is a standard, companies know that it will not change. Not only for the standard, but basically, because the technology was mature enough in 2012 to be incorporated in companies' hardware. Since 2012 it hasn't changed much. Few updates of specs, some improve[ment]s but it is quite stable in all implementations of devices. There are new things since 2012 but not in releases in WR devices. The protocol of the switch is not changing." CH6 "The changes compared to the V3.4 switch fall into two categories: replace existing components due to obsolescence, and changes to allow the easy installation of a low-jitter-daughterboard."(WR repository, 2020) "The backplane has been modified so that the fans are always on (in the original backplane 3.3 design the fans are controlled by software). This keeps the hardware cooled in all circumstances" (WR repository, 2020).
Development	"What we want to try with proprietary implementations is to meet the requirement of picosecond accuracy, but not with custom hardware as we want to make it more generic—and removing all our infrastructure would not make much sense." CH5 "The way you do an open source on software, you tend to get it to be implemented in different microprocessor architectures. However, hardware is different. So, it's like any time you put two pieces of hardware together, they are never going to be exactly the same. Therefore, as WR evolves, the real challenge is that they have tried hard to make it such that you just push a button and things are configured automatically and pull in the relevant files for architecture, but it does not work like that. It is not easy; this is really hard." CH6 "The main limitations of WR is that it is a brand-new hardware design, a custom hardware. It does not have the off-the-shelf components and we should develop things to overcome these limitations." CH1
Physical instantiations	• Frankfurt Stock Exchange's deployment of WR • Vodafone proof of concept of WR in the Netherlands • D-TACQ Solutions Ltd. proprietary WR • Picoquant proprietary WR • SyncTechnology proprietary WR