Journal of the Association for Information Systems

Volume 26 | Issue 6 Article 8

2025

Layering the Architecture of Digital Product Innovations: Firmware and Adapter Layers

Julian Lehmann
Arizona State University, j.lehmann@asu.edu

Philipp Hukal
BI Norwegian Business School, philipp.hukal@bi.no

Jan Recker
University of Hamburg, jan.christof.recker@uni-hamburg.de

Sanja Tumbas
Bern University of Applied Sciences, sanja.tumbas@bfh.ch

Recommended Citation

Lehmann, Julian; Hukal, Philipp; Recker, Jan; and Tumbas, Sanja (2025) "Layering the Architecture of Digital Product Innovations: Firmware and Adapter Layers," *Journal of the Association for Information Systems*, 26(6), 1630-1650.
DOI: 10.17705/1jais.00956
Available at: https://aisel.aisnet.org/jais/vol26/iss6/8

Journal of the Association for Information Systems (2025) **26**(6), 1630-1650
doi: 10.17705/1jais.00956

RESEARCH ARTICLE

ISSN 1536-9323

Layering the Architecture of Digital Product Innovations: Firmware and Adapter Layers

Julian Lehmann,[1] Philipp Hukal,[2] Jan Recker,[3] Sanja Tumbas[4]

[1]Arizona State University, U.S.A., j.lehmann@asu.edu
[2]BI Norwegian Business School, Norway, philipp.hukal@bi.no
[3]University of Hamburg, Germany, jan.christof.recker@uni-hamburg.de
[4]Bern University of Applied Sciences, Switzerland, sanja.tumbas@bfh.ch

Abstract

This study investigates how organizations layer their product architectures by embedding digital components into physical products. Drawing on a longitudinal case study of PrintCo—a desktop 3D printer firm—we show that layering a product architecture relies on creating adapter layers that facilitate connections among physical and digital components. To generate these adapter layers, PrintCo first parametrized physical components through firmware, making them controllable and addressable. PrintCo then arranged higher-order digital functionality via adapter layers that couple parametrized physical components with additional digital functionality. Based on these findings, we propose a theoretical model that explains how organizations layer product architectures, what the role of adapter layers is, and how the transformation of an organization's product architecture progresses.

Keywords: Digital Product Innovation, Firmware, Embedding, Layering, 3D Printing, Case Study

Dorothy E. Leidner was the accepting senior editor. This research article was submitted on January 19, 2023, and underwent four revisions.

1 Introduction

With the advent of digital technology, organizations are transforming a growing number of industrial-age physical products—from home appliances (Henfridsson et al., 2018) and cars (Hylving & Schultze, 2020; Svahn et al., 2017) to manufacturing technologies (Sandberg et al., 2020) and buildings (Wang et al., 2022)—into digital product innovations, that is, new combinations of physical and digital components (Yoo et al., 2010). Digital product innovations exhibit a unique product architecture [1] (Yoo et al., 2010) of functionally abstracted *modules* (Baldwin & Clark, 2000; Hylving & Schultze, 2020; Pujadas et al., 2024) arranged in separate, loosely coupled vertically stacked *layers* that bundle functionally related sets of components that serve a particular purpose (Yoo et al., 2010).

A layered-modular product architecture has several benefits: Modularity instills products with variability and flexibility because organizations can adjust or replace individual components with more powerful ones without affecting the product as a whole (Albert & Siggelkow, 2022; Colfer & Baldwin, 2016). Layers separate more static components with fixed, unchanging functions (like most physical parts) from more flexible ones (often digital parts), which allows organizations to continuously adapt their products (Faulkner & Runde, 2019; Yoo et al., 2010).

[1] A product architecture refers to the set of components and their interactions that constitute a product (Henderson & Clark, 1990).

While prior research has investigated how to modularize a product architecture, much less is known about how organizations layer product architectures. Existing research points out that layering product architectures involves the design and integration of digital representations—depictions of real-world information, objects, or phenomena encoded in a digital format that can be processed, stored, or transmitted by computers—into product architectures (Lyytinen, 2022). However, doing so is a complex task that goes beyond merely grafting digital technology components onto physical products (Hylving & Schultze, 2020; Lyytinen, 2022). Instead, organizations must embed digital technology components within a product architecture (Lyytinen, 2022) such that these digital components become the primary means for controlling physical components and the basis for introducing additional digital functionality leveraging the physical components. This process poses a substantial challenge to organizations because even if modular, physical components are rarely designed to interact with digital technologies. Therefore, our research objective is to understand how organizations layer their product architectures by embedding digital components.

We conducted a longitudinal inductive case study of PrintCo, a fused deposition modeling (FDM) desktop 3D printer company that layered its printers' product architecture to meet evolving user needs in terms of accuracy, reliability, and the capability to print increasingly complex objects. When PrintCo initially launched its 3D printers along with complementary CAM software as its market offering, these components were originally decoupled and existed as separate systems. When PrintCo decided to enhance the product's performance to be more attractive to its users through software functionality, it needed to embed new layers of digital components within the printers' product architecture.

Through our analysis, we uncovered that PrintCo relied on two techniques to embed new adapter layers within the architecture of its 3D printers. First, *parametrizing physical components* involves the iterative creation of digital representations of physical components to capture their attributes and primary functions in digital form. These make physical components controllable through digital code. Second, *arranging digital functionality* involves establishing digital adapter layers within a product architecture to couple higher-order digital functionality with lower-level digital representations. Together, these two techniques explain how organizations embed digital technology components within a product architecture to create a layered-modular product architecture.

Our study makes two main contributions. First, our findings extend our knowledge about digital product architectures (Colfer & Baldwin, 2016; Lee & Berente, 2012; Svahn et al., 2017). Specifically, we suggest that organizations layer their product architectures by separating functional layers that bundle related product capabilities from adapter layers that organize the interaction between these capabilities. As we highlight, the locus of layering thus follows a logical hierarchy of steps that successively ascend a product architecture. Second, we provide new insights into the techniques organizations can take to layer their product architectures (e.g., Hylving & Schultze, 2020). We propose that layering a product architecture involves two key techniques: parametrizing physical components and arranging digital functionality. Both techniques help organizations operationalize the goal of layering their product's architecture into concrete design decisions they can implement.

Our paper proceeds as follows. First, we ground our study in the literature on digital innovation and product development to establish the concepts of modularity and layeredness. Then, we describe the procedures of our field study and present the findings from our inductive analysis. We then discuss the theoretical insights that flow from our study before reviewing the implications and limitations.

2 Background

2.1 Modularity and Layeredness of Digital Product Innovations

In their seminal work, Yoo et al. (2010) pointed out that digital product innovations have a distinctive layered-modular architecture with three key properties. First, a layered-modular architecture is structured. It separates layers of digital components (such as software routines, algorithms, or data) from the physical layers on which they reside. This makes digital components product-agnostic (Nambisan et al., 2017; Yoo et al., 2010): Their design requires minimal consideration of eventual use or the specific product architecture they will be part of (Eaton et al., 2015; Garud et al., 2008). Second, a layered-modular architecture is malleable (Nambisan et al., 2017; Yoo, 2010): It can be equipped with new computing instructions to perform fundamentally new functions at any point in time. Third, due to its malleability and ability to be addressed by other computing devices, a layered-modular architecture can support open-ended, continuous change cases (Huang et al., 2022; Yoo et al., 2010; Zittrain, 2006) that can be driven by large and uncoordinated groups of third-party contributors (Gawer, 2021; Parker & Van Alstyne, 2018).

Instilling product architectures with these properties requires organizations to layer their product architecture even when it is already modular (Yoo, 2013). However,

precisely *how* organizations can layer the architecture of their products remains largely unexplored. Past work has focused mainly on the implications of a layered-modular architecture (Baskerville et al., 2020; Henfridsson et al., 2014; Yoo, 2010), or it has focused on organizational challenges, such as tensions between different organizing logics (Svahn et al., 2017), growth (Giustiziero et al., 2023; Huang et al., 2022), new organizational forms and value creation (e.g., Henfridsson et al., 2018; Lorenz et al., 2024; Parker et al., 2017), or the design of interfaces as a key vehicle for addressing and controlling product components (e.g., Gawer, 2021; Kuan & West, 2023; Pujadas et al., 2024). In the pursuit of advancing our understanding of digital innovations, this work takes the layered-modular product architecture as given, despite evidence suggesting that layering a product architecture involves an intricate set of challenges (Hylving & Schultze, 2020). At the core of these challenges is the requirement to decompose a product architecture into sets of hierarchically structured, functionally related components, i.e., layers.

Building on the work by Simon (1996) and Baldwin and Clark (2000), innovation management researchers suggest that products can be understood as a hierarchy of loosely coupled subsystems—modules—that interact via standardized interfaces. In theory, the decomposition of a product into loosely coupled modules enables changes to individual modules without affecting other parts of the product. This decomposition allows organizations to create product variants flexibly (Huang et al., 2022), such as a scoped-down, low-cost version, or to "open up" their products to third parties that enhance the performance of the product by providing specialized modules (Baldwin, 2023; MacCormack et al., 2006).

Organizations can modularize a product architecture through an iterative process, referred to by Baldwin and Clark (2000) as "design rationalization." This process mainly involves identifying interdependencies among components and gradually adding standardized interfaces that define "design rules" for how the components should interact (Baldwin, 2023; Baldwin & Clark, 2000). Design rules reduce the need for control and direct coordination as long as engineers adhere to them (Gawer, 2021; Kuan & West, 2023). In this sense, modular architectures emerge from the continuous specialization of components in a product architecture to form and integrate modules via the definition of design rules.

While this work is instructive for modularizing product architectures, it is not directly applicable when organizations seek to *layer* product architectures (Hylving & Schultze, 2020; Lyytinen et al., 2016). Layering means organizing product components into vertically stacked and functionally related sets of components, each representing distinct stratums within a product and implementing specific sets of functionalities. Higher layers build upon the capabilities provided in lower layers.

Thus, layering differs from modularizing a product architecture in at least three ways (Hylving & Schultze, 2020): First, modularizing means breaking down a product into subsystems, while layering means organizing product components into at least two distinct sets of functionally related components that may be more or less modular. The components within each layer collectively (rather than individually) fulfill a dedicated purpose (e.g., data transmission vs computation) as part of the overall product. Second, layering is not primarily concerned with how functions are allocated to individual modules within a layer but with how the different layers interact to form a cohesive, integrated product architecture. Third, a layered architecture is strictly hierarchical and unidirectional (Hylving & Schultze, 2020): Higher layers build and act upon lower layers, not vice versa. These distinctions highlight that successful digital product innovation requires not just a commitment to modularity but also to layering to ensure that each component aligns and integrates effectively within the overall product. Table 1 summarizes the key concepts that inform our study.

Table 1. Key Concepts From the Literature That Informs Our Empirical Study

Concept	Design intent	Main techniques used	Main outcome
Modularizing	To achieve functional abstraction of component	Encapsulation and interface design (Baldwin, 2023; Brusoni & Prencipe, 2001).	Abstraction of modules that hide internal logic and which can be accessed through interfaces.
Layering	To organize sets of semantically distinct components within stacked	Embedding sets of functionally related digital components within a product architecture, thus establishing separate layers (Hylving & Schultze, 2020; Tilson et al., 2010).	Separation of stable from fluid product components to allow for generative performativity and extension.

2.2 Embedding Digital Components into Product Architectures

Lyytinen (2022) suggests that layering product architectures requires organizations to embed digital components within them such that all components can be connected to an emerging array of digital service layers. Embedding digital components within product architectures involves capturing real-world phenomena (e.g., social interactions, material performances, commercial transactions) in digital form as well as adding computing hardware (i.e., microcontrollers, sensors, actuators) to a product architecture such that other layers of digital technology can act upon them (Hylving & Schultze, 2020; Lyytinen, 2022).

Yet organizations seeking to embed digital components into product architectures face at least two key challenges. The first challenge concerns how aspects of the real world can and should be represented in digital form. Digital product innovations are cultural objects (Alaimo & Kallinikos, 2022) and always occupy a social position (Faulkner & Runde, 2019). There is no objectively correct way to represent real-world phenomena in digital form. Additionally, digital representations do not exist in a vacuum; instead, they must connect to and fit within existing institutional arrangements to be useful (Lehmann et al., 2022). For instance, a Bluetooth speaker must adhere to the interface specifications of a transmitting device to receive and process audio signals. These signals must be provided in digitally encoded form for the speaker's output to be recognizable as music or speech.

The second challenge concerns how organizations can embed digital technology components within a product architecture. This challenge exists because physical and digital components are not the same (Faulkner & Runde, 2019): Digital technology components are collections of bitstrings that can be rearranged and manipulated to alter form and function dynamically, while physical components bind form and function in a stable configuration (Henfridsson et al., 2014; von Briel et al., 2018). This implies that rearrangements of product components are inevitable when organizations try to embed digital technology components within product architectures. For example, while digital innovation researchers suggest that digital product innovations can be reprogrammed with minimal consideration of underlying physical components (Yoo et al., 2010), instilling products with this property means enabling an emerging set of interactions among product components (Sandberg et al., 2020). This, however, is difficult because component interactions are typically fixed and thus hard to change, as per their design rules (Baldwin & Clark, 2000).

Moreover, although digital components may, in principle, be product-agnostic (Nambisan et al., 2017; Yoo, 2013), digital product innovations also contain purpose-built physical components beyond their computing machinery. For example, in contrast to general-purpose von Neumann architectures, 3D printers are designed for a specific purpose—printing objects using polymer filament—and can hardly be made to perform other functions (Lyytinen, 2022). Together, this suggests that digital components may not be quite as product-agnostic as often assumed (Henfridsson et al., 2018): They must always be coupled with physical components to operate within a specific physical context to be meaningful and valuable (Goebeler et al., 2024).

Taken together, layering a product architecture requires organizations to embed digital technology components within a product architecture that can (1) trigger the execution of code, (2) represent physical and social real-world phenomena in digital form so that they can compute on them, and (3) contextualize digital computations within the real-world context in which they are used (Lyytinen, 2022). Layering product architectures thus goes above and beyond merely adding a computing device to a product or developing a software application that can be connected to a product. However, how organizations embed digital components into physical products to layer them remains unclear.

3 Method

We engaged in a form of grounded theorizing (Strauss & Corbin, 1998; Urquhart et al., 2010) based on a longitudinal single-case study of an FDM desktop 3D printer manufacturer (PrintCo). We collected data as representative facts (Sarker et al., 2018) about how PrintCo transformed its product to integrate digital and physical components. Our data analysis strategy was inductive and involved abstracting from occurrences to events (Abbott, 1990), temporal bracketing (Langley, 1999), and inductive coding (Strauss & Corbin, 1998). Because our interest was in understanding the techniques through which PrintCo transformed the architecture of their 3D printers, we decided to draw on the literature on digital product innovation and technology and innovation management as lenses (Henderson & Clark, 1990; Yoo et al., 2010).

3.1 Research Setting

PrintCo is a leading firm in the desktop FDM 3D printing industry. Our analysis of PrintCo builds on data from six years (2011-2016), during which PrintCo designed and introduced several increasingly sophisticated 3D printers. In 2016, PrintCo launched a new 3D printer product line that integrated numerous digital components—a stark contrast to the product lines we studied.

In general terms, the architecture of a desktop FDM 3D printer consists of several components that work

together to create three-dimensional objects from digital models of physical objects by processing polymer plastic materials. Computer-aided manufacturing (CAM) software translates digital models into machine code instructions to produce a physical object from a digital model. This process involves "slicing" digital model files into individual layers, resulting in a G-code file. This G-code file contains machine instructions for each layer. The G-codes generated by the CAM software are then transmitted to the 3D printer, which translates these codes into electronic signals that control the physical components.

Of all the physical components, the *frame* is the structural backbone of a 3D printer, providing stability and support for all other components. It ensures that the printer remains rigid and reduces vibrations, which is crucial for maintaining print quality. The *print bed* is the surface on which the 3D-printed object is built. The *extruder* is responsible for feeding and melting the filament and then depositing it layer by layer through the *nozzles*. Nozzle sizes can vary, affecting the level of detail and speed of prints. *Stepper motors* drive the movement of the printer and control the extrusion of filament. The typical motors are an *x*-axis and *y*-axis motor moving the print head horizontally, a *z*-axis motor moving the print bed vertically, and extruder motors driving the filament into the hot end. Belts and rods transfer the motion from the motors to the print head and bed, ensuring precise movements. Belts are typically used for *x* and *y* movements, while threaded rods or lead screws are used for the *z*-axis. *Sensors* detect the limits of the printer's movements in the *x, y,* and *z* directions, ensuring that the print head does not move beyond its intended range. A *control board* interprets the G-code (instructions from the slicing software) and controls the motors, heaters, and sensors accordingly. The control board also receives feedback from the sensors to ensure accurate positioning of the print head and bed. These components must work in harmony for the 3D printer to function.

We consider PrintCo a revelatory empirical case because the challenge of how organizations layer a product architecture was particularly salient. Digital technology, such as computer-aided design and manufacturing (CAM) software, has always been crucial in FDM 3D printing (Rayna & West, 2023; West & Kuk, 2016). PrintCo's early market offering consisted of two largely separate systems, the physical printing device on the one hand and CAM slicing software on the other. Over time, PrintCo tried to improve the printer's reliability and accuracy to meet users' evolving needs by embedding digital technology directly within the printer's product architecture. Table 2 summarizes the main product generations' technical characteristics that we studied.

3.2 Data Sources

We used five primary data sources (Table 3). We conducted 30 semi-structured and five informal interviews with key informants in two waves between 2017 and 2019. While our questions were initially broad, we refined them as interviewing progressed. Initially, we asked informants about which new digital technology components PrintCo introduced to extend the functionality of its printers. Later, we asked them why these digital technology components became more critical to how PrintCo sought to create value for its users, as well as how PrintCo managed to integrate digital technology with the other components of the product's architecture. In addition, we collected archival data consisting of around 300 company-internal documents, five secondary interviews, and around 190 publicly available documents. This data provided detailed insights into the actions taken by PrintCo to transform its 3D printers.

Table 2. Characteristics of the Different Generations of PrintCo's Printer

Year after founding	1	3	4	6
Main physical components	Self-assembly kit (plywood) with single extruder, design based on the RepRap project	Preassembled assembled, single extruder; enclosed build chamber, heated bed upgrade	Improved feeder system, interchangeable nozzles, single extruder	Dual-extrusion system, auto-bed leveling, swappable print nozzle system, advanced cooling system
Main digital components	Decoupled and generic slicing tool wrapped in a custom GUI	Revised, proprietary slicing engine for enhanced performance	Enhanced with ways of adapting print process to user needs	Continuous improvements, implemented GUI APIs for integrating functional extensions
Main firmware	Generic open-source firmware with basic parameters for controlling printer components	First custom firmware implementing broad range of custom machine codes	Firmware enhancements for improving print speed and accuracy	Advanced firmware with dual-extrusion support and remote control, including generation of print reports
Interfaces	USB	USB, SD card	USB, SD card	Wi-Fi, Ethernet, USB, APIs

Table 3. Data Sources

Data sources	*N*	Description	Temporal coverage
Semi-structured interviews	30	Interviews with key employees involved in research and development, product management, strategy, and portfolio management of 3D printers. Interviews include initial and follow-up interviews	2012-2016
Informal interviews	5		
Secondary interviews	5	Interviews with PrintCo's top management in the popular press and industry publications	2012-2016
Public documents	~190	Systematic collection of documents from sources including company blog entries, forum posts, press releases, product change logs and release notes, and technical documentation	2012-2016
Company documents	~300	Documents informing about product strategy, specifications of new machines concerning both digital and physical components, development process, project management, deliverables, and milestones	2014-2016

3.3 Analysis Strategy

Our data analysis followed a process approach (Berends & Deken, 2021; Langley, 1999) to trace the unfolding of key events in PrintCo's journey of improving their 3D printer.

We first compiled a detailed event list of how the 3D printers' overall product architecture changed, based on archival data and company documents (release notes, blog posts, product documentation, forum posts, presentations). We focused broadly on changes that were made to physical and digital components and the relationship between the components featured in the different product generations. We defined as salient events (Abbott, 1990) all those actions PrintCo undertook that related to designing, revising, changing, and updating digital and/or physical components. Examples included the implementation of new machine code instructions for obtaining data from a new sensor and the design of algorithms for using one print nozzle to generate support structures. By contrast, events such as the appointment of a new C-level executive or the creation of a new department did not qualify as events salient to our study and were thus excluded from our analysis. Each event was marked with a timestamp and accompanied by detailed information about the event to maintain a chain of evidence. We identified 156 such events in total.

Next, we analyzed our data to glean insights into the sequential and logical unfolding of these events, considering the objectives pursued by PrintCo at that time. For example, from company-internal documents, we learned that PrintCo's key ambition initially was to make its 3D printer increasingly attractive to users by integrating digital and physical components. We noticed that to realize this vision, PrintCo had to integrate digital technology with physical components such that it became possible to digitally resolve the physical shortcomings of its printers (e.g., warping of printed objects) or to increase the versatility of the 3D printers (e.g., printing layers with varying widths). Through temporal bracketing (Langley, 1999), we then clustered salient events into nine logically, thematically, and temporally related *episodes* (Table 4). Each episode had a distinct objective, such as making printers more reliable and accurate or extending the ability to print more complex objects.

We then treated the episodes as embedded units of analysis and thematically coded interviews and company documents to identify the key activities within and across the episodes that explained how PrintCo embedded digital technology components into its product's architecture. In doing so, we noticed recurring patterns across episodes. Specifically, we learned that PrintCo started implementing additional digital layers whose primary purpose was to couple functional digital and physical components. We labeled these layers "adapter layers" because they helped two separate layers to communicate with one another and thus established sets of components as layers. Overall, we synthesized two distinct techniques by which PrintCo created these adapter layers—*parametrizing physical components* and *arranging digital functionality*.

Finally, having identified these techniques, we examined their temporal and logical relationships. We found that PrintCo began by parametrizing physical components and only afterwards moved to arranging digital functionality. That is, the organization initially set up a digital adapter layer (firmware) embedded within their physical components before integrating higher-order digital components with other digital elements through additional adapter layers. Figure 1 presents our data structure as an analytical ladder (Gioia et al., 2013; Urquhart et al., 2010), progressing from first-order codes (e.g., introducing parameters) to second-order themes (e.g., designing digital representations) and aggregate dimensions.

Table 4. Main Product Development Episodes

#	Start	End	Episode name	Description	Main objective	Key events
1	Q2/2011	Q2/2012	Control panel	Augmentation of printer architecture digital display to control printer functions without software	Provide a new user interface for controlling the printer during a print job	1. Introduce an LCD control panel for controlling the printer, along with an SD card interface. 2. Design codes for reading from and writing to SD cards. 3. Leverage the control panel to give users access to machine settings during the printing process. 4. Release firmware update for operating the control panel.
2	Q4/2012	Q3/2014	Heated bed	Redesign of print bed to be heated	Resolve issues related to the warping of objects	1. Redesign print bed component to incorporate heating unit. 2. Revise print bed representation to include a parameter for temperature. 3. Make parameters available in slicing software to set the print bed temperature.
3	Q3/2013	Q4/2016	Dual extrusion	Redesign of print head introducing a second nozzle	Provide the option to print in two materials or colors	1. Develop a prototypical dual-extrusion upgrade kit initially so that users can upgrade their existing printers. 2. Implement new machine codes for dual-extrusion functionality, such as wiping nozzles or cooling down an inactive nozzle. 3. Revise the design of the dual-extrusion print head, heating, and nozzles to address shortcomings of the first kit. 4. Revise firmware and slicing software with advanced functions and support for dual-extrusion printing.
4	Q1/2014	Q4/2016	Active bed levelling	Redesign of the print bed allowing for automated height adjustment	Increase the reliability of the printer	1. Design a new capacitive sensor that can be used to measure the distance between the nozzle and print bed. 2. Implement machine codes for reading sensor data, accounting for noise during measurement and failed measures. 3. Implement machine codes for automatically adjusting the vertical position of the print bed. 4. Revise slicing software to incorporate print bed calibration before printing to improve print outcomes.
5	Q2/2012	Q1/2013	Stand-alone printing	Design of software routines to establish the printer as a standalone device without requiring a connection to a PC	Establish professional workflow	1. Design a new feature in slicing software to generate and export G-code files that rely on the ability to print from an SD card. 2. Decommission feature to send machine instructions to printer directly via cable in the slicing software.
6	Q3/2014	Q3/2016	First-layer adhesion	Design software routines to improve the adhesion of the first printed layer to the print bed	Increase the success rate of print jobs that would otherwise fail due to poor adhesion of the first layer	1. Optimize slicing settings using the heated bed functionality and sensor information about the distance between the print bed and print head. 2. Design new functionality for generating skirt and brim structures to be printed around the actual model to increase adhesion due to a larger footprint. 3. Optimize the design of these structures depending on model and material properties.
7	Q3/2015	Q4/2016	Tilt correction	Design of new firmware routines that account for tilted print beds	Improve the accuracy of printed objects	1. Implement the ability to measure the distance between the print bed and print head to determine the tilt of the print bed. 2. Adjust slicing software to account for any tilt.
8	Q3/2016	Q1/2017	Complex objects	Design of new software routines that improve printing of objects with intricate details and overhang	Extend the range of use cases of its 3D printers	1. Leverage dual-extrusion print head by introducing the ability to select a specific nozzle in slicing software. 2. Design a new feature for decomposing a model into separate parts, each to be printed with one model. 3. Design a new algorithm for generating support structures that enable printing particularly complex or difficult-to-print objects. 4. Explore the integration of support materials that can be removed or resolved in water.
9	Q1/2016	Q1/2017	Materials	Improve print outcomes with materials	Support a larger array of use cases	1. Explore ways to adapt print settings. 2. Devise settings to optimize print process for intended use.

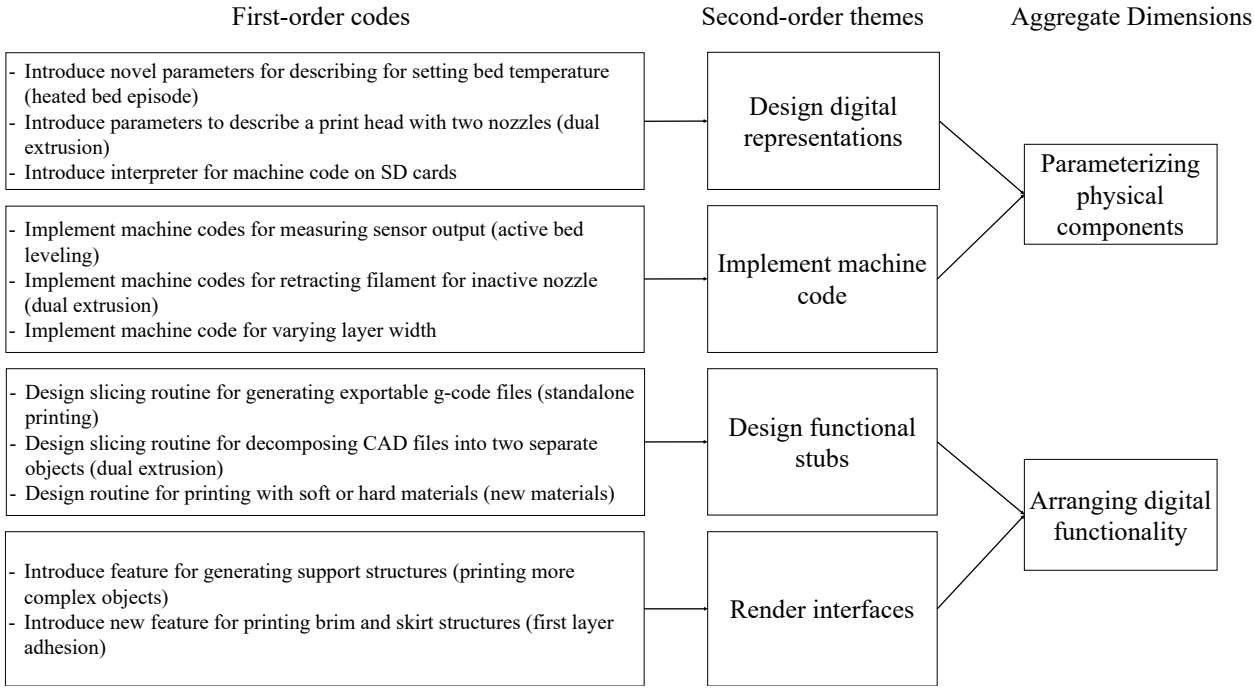

Figure 1. Data Structure

4 Findings

4.1 Overview: How PrintCo Layered Its Product Architecture

PrintCo was founded in the early 2010s with the vision of designing 3D printers for and with a community of makers and tinkerers. PrintCo introduced its first 3D printer as a self-assembly DIY kit. Shortly after the release of the first printer, PrintCo assumed stewardship of an open-source slicing software as a means for operating its printers and to make 3D printing accessible to a broader audience, which led to faster adoption among makers and tinkerers. Over time, users became interested in printing increasingly complex and sophisticated objects, with several using their printers for more advanced applications. In noticing this trend, PrintCo worked on performance improvements to meet these evolving user needs by integrating digital and physical components. As a software architect summarized:

> *We're not doing any of the steps necessarily better than any of the others, but we are doing all of the steps. And I've seen a lot of 3D printers that either had an amazing bit of hardware, but the software was unusable, slow, nobody understood it. Or the other way around: you have a very crappy machine and then the software was very easy to use. And I think that's why we are pretty successful, you know, we had sort of the magical combination of a pretty decent machine with pretty decent software.*

This focus impacted the relationship between the digital slicing software and the printer. Initially, the slicing software and printer were decoupled, largely independent, and existed as separate systems. The slicing software was internally viewed as a "side project" that was built off a generic G-code interpreter with minimal customization. To better meet user needs, software became a key driver for improving printer accuracy and reliability and enhancing usability across an expanding range of applications.

To achieve this goal, PrintCo aimed to integrate the slicing software into the printer's product architecture to enable new features and improvements. For instance, PrintCo continually improved its slicing software to offer a larger number of increasingly powerful functionalities that would make 3D printing more attuned to user requirements in accuracy and consistency, they made the slicing algorithm more robust and accurate, and they designed new algorithms for printing increasingly complex objects (e.g., with lots of overhang). However, this process was far from trivial. As PrintCo sought to introduce additional functionality via software, they first needed to find a robust way to controll physical components via digital code. At the same time, PrintCo also needed to ensure that primary functions offered by the components of the product architecture could be implemented by the slicing software.

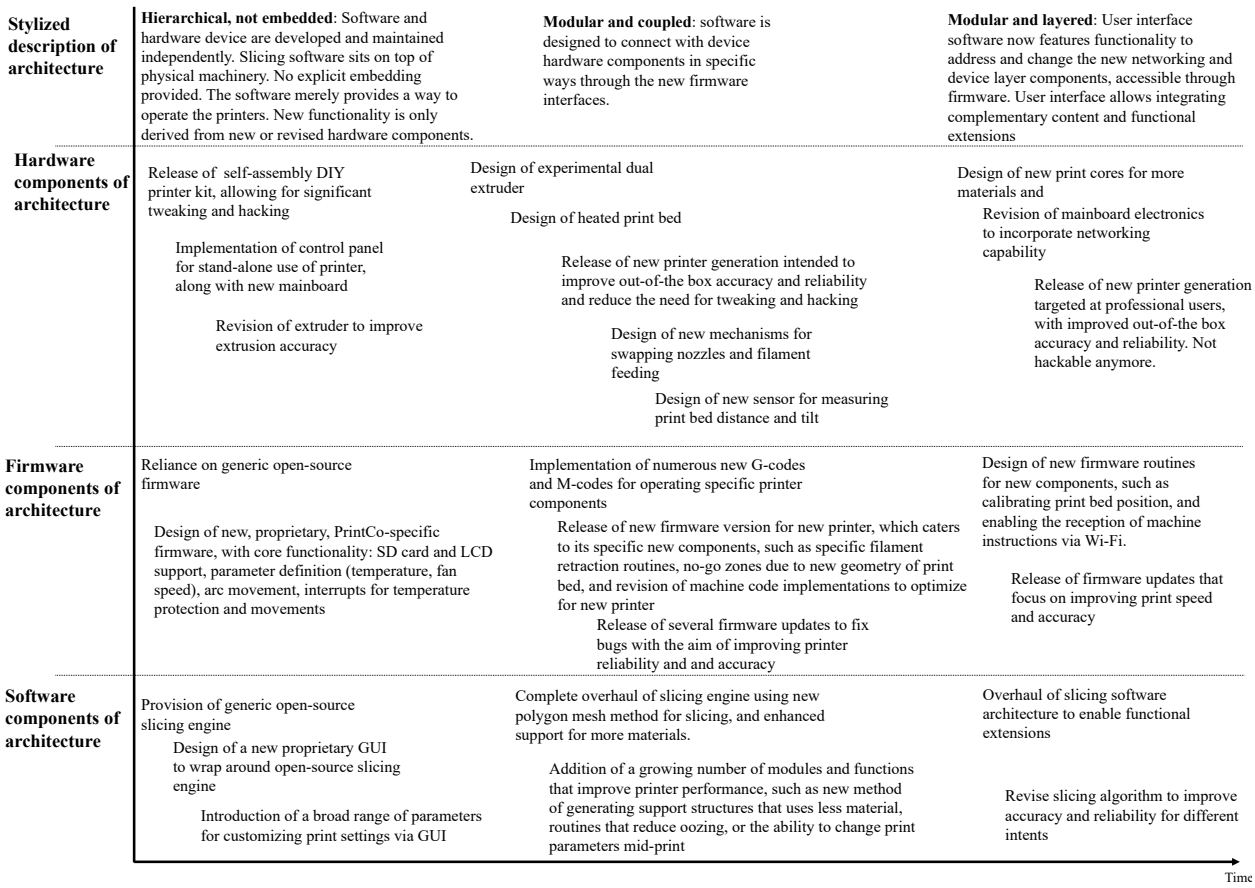

Figure 2. Overview of the Architectural Changes in PrintCo's 3D Printer Over Time

Figure 2 summarizes the evolution of the architecture of PrintCo's 3D printer over time, offering a stylized description at three different stages and differentiating changes between hardware, firmware, and software components (with indented entries representing temporally subsequent steps). Through our analysis, we uncovered two techniques through which PrintCo layered its product architecture. First, PrintCo sought to enable control of physical components through digital code. We label this technique *parametrizing physical components* to describe how PrintCo represented physical components in digital form within a firmware layer such that these components' primary functions (such as rotating the print head or heating the print bed) could be addressed by yet-to-be digital layers. Second, PrintCo also layered its product architecture by *arranging digital functionality* within adapter layers that became part of the product architecture. These adapter layers allowed PrintCo to couple multiple functional digital components across layers and enable connections to further digital components. Through these two techniques, PrintCo embedded adapter layers within its printers' architecture. These layers facilitated interactions among functional components and organized the architecture into digital and physical layers.

4.2 Parametrizing Physical Components

The first technique through which PrintCo layered its product architecture was *parametrizing physical components*. Through this technique, PrintCo made the primary functions of physical components addressable through digital code. Parametrizing physical components advanced an emergent firmware layer—sitting atop and abstracting from the underlying physical components—within the product architecture. Firmware traditionally consists of a fixed set of instructions and routines for operating physical components. However, our analysis revealed that firmware played a crucial role as a dynamic adapter layer to make the physical components' primary functions digitally accessible such that they could be loosely coupled with and flexibly operated by yet-to-be-developed software layers higher in the architecture. These software layers turned out to be key for equipping the printer with additional functionality.

Beginning with physical components, the parametrization of components involves two crucial activities. First, to ensure physical components can be controlled via digital code, PrintCo created digital representations of physical components in the form of parameters, variable ranges, and standard settings that comprehensively described these components. These digital representations modeled

the behavior and state of underlying physical components, and captured key values associated with the various states the components could be in to ensure that the firmware could interact with the hardware and adapt its behavior in a structured and efficient manner. The purpose of the digital representations was to convey the relevant static attributes of physical components, such as the axes along which a mechanical element could move or the color palette a screen could display. They defined the state space of physical components within an emerging adapter layer of the printer (i.e., firmware). Thus, creating these representations required PrintCo to determine which parameters were crucial for effective control, and defining these representations was essential for transitioning from an arrangement where component roles and interactions were fixed to a more adaptable arrangement.

Second, PrintCo implemented primary functions that a physical component could perform (e.g., moving and turning a print head along certain axes, or extruding filament through a feeder in a particular mass and pressure) in the form of *machine code instructions*, so-called G-codes and M-codes.[2] Essentially, machine codes define when and for how long electricity should flow through which pins on the printer's microcontroller to have a component perform a certain function and thereby transition from one component state to another. Machine codes were implemented in C++ code and compiled to run on the printer's microcontroller. By implementing machine codes, PrintCo added dynamic elements to an emerging firmware layer to control the printer components. Thus, parameterizing physical components advanced a digital adapter layer within the product architecture that provided the means for the firmware to control the behavior of printer components. PrintCo iterated between the parametrization of physical components and the implementation of machine codes to enable the subsequent introduction of new digital functionality.

PrintCo's implementation of a heated print bed provides an apt illustration for parametrizing components through digital representations and machine code instructions. A known issue that stood in the way of enabling a larger set of use cases was shortcomings in print quality, such as warping (printed objects shrank and became asymmetrical when they cooled down too quickly). Warping could lead to suboptimal or even failed print outcomes. It was, in fact, a common practice among users to redesign objects with a wider base to minimize warping.

PrintCo addressed this issue by redesigning the print bed—a physical component—so it could be heated. PrintCo based the design of a heated bed on the existing design, exploring ways to integrate a heating element that could be powered by the printer's existing power supply unit yet was powerful enough to reduce the temperature difference between filament extruded from the print head and the print bed enough to prevent warping. PrintCo conceived of the heated bed as an improved substitute for the existing print bed to address problems with warping objects and failed prints.

The heated print bed promised to resolve issues due to warping and improve print quality. But this required the print bed temperature to be adjustable to fit varying requirements and to be controlled digitally. PrintCo therefore implemented in the firmware a digital representation of the component's parameters and possible states that described the key properties of the heated print bed as well as the code to initialize the heated bed when turning on the printer. These actions added to the static elements of the emergent firmware layer, which ensured the component's new features could be implemented in the actual printing process. Subsequently, PrintCo implemented machine codes in its firmware to introduce the ability to heat the print bed to certain temperatures by obtaining and setting the printer dynamically from a future, higher layer of application software. As summarized in an R&D document:

> *To develop the heated bed, the following has been done: The prototype has been designed, cables have been designed and tested, a heated bed has been designed and tested, the firmware was adjusted so that the new product can be chosen and a good experience will materialize.*

These efforts ensured that the heated print bed could be embedded within the printer's product architecture to enhance print quality.

The dual-extrusion print head provides a second illustration for parametrizing components. Dual extrusion refers to the ability to print with two different material types, colors, and/or widths simultaneously. Shortly after introducing its first-generation 3D printer, users became interested in printing more complex objects—objects with intricate geometrical details. For instance, printing objects with overhang carried the risk of an object collapsing. Since such objects were inherently challenging to print with single-extrusion FDM 3D printers, PrintCo therefore initiated the design of a dual-extrusion print head to replace the existing single-extrusion print head.

PrintCo implemented the dual-extrusion functionality through a multistep process that began with the design of a prototypical print head, which PrintCo released as an experimental upgrade kit to users. Through this step, PrintCo learned that, although the new print head was designed to replace the existing one, the dual-extrusion

[2] G and M codes are alphanumeric instructions used in manufacturing to control and automate machine movements and operations.

print head changed how the printer had to execute print operations for dual extrusion to work effectively. As an engineer told us:

> *If you have two nozzles at the same height and you deposit material, the material swells up a little bit after the nozzle has deposited it. But if you have two nozzles, you can rock over the layers that you previously put down, as well as the second nozzle that is idling may be leaking material onto the model. The initial thermal design for the print head was insufficient for dual extrusion.*

These insights highlighted the need for further modifications to optimize dual extrusion and prevent unintended material deposition.

In response, PrintCo's engineers revised the digital representation of the print head through additional parameters within the firmware layer. This representation described the print head's various attributes and possible states, such as which nozzle was active and extrusion temperatures for each extruder, thus adding to the firmware's static elements. Subsequently, PrintCo implemented machine codes that could operate and control nozzle lifting and filament extrusion from two nozzles. One specific challenge in parametrizing this component was to ensure that the inactive nozzle did not ooze. PrintCo did so through the implementation of a sophisticated cooling mechanism in its firmware to solidify the material in an inactive nozzle, thus ensuring that material did not ooze from the print head:

> *We never developed something that really shut off the second nozzle from leaking. That was solved in software alone by determining, at a certain temperature, that the material inside the nozzle was solidifying so much that it would not leak out anymore. The software team designed a strategy that once you've reached the end of a move and the end of the use of the second nozzle, you turn off the power, so the material will not leak as much. … That's something, you can try and solve in hardware, but it's super hard, in case you need to really find a solution to close off a very tiny hole of 0.4 millimeters and make sure it's repeatable and clean and every time, and in software, it's by implementing that strategy, it's a lot easier to solve.* (Engineer and manager)

This software-based solution not only addressed the oozing issue effectively but also demonstrated the advantages of leveraging firmware to control hardware.

While PrintCo was eager to support dual extrusion and had developed early versions of functions for operating a dual-extrusion print head, the initial design of the dual-extrusion print head evoked new interdependencies due to different geometrical properties compared to a single-extrusion print head. Simply introducing the new component did not only not yield the desired outcomes; it deteriorated the performance of its printers. A firmware engineer reflected on this design:

> *You have one print head and you have two nozzles, and the nozzles, if you manufacture them, are always slightly different, and you need to adjust it manually. I could print very well, but I was very skilled in doing it. For people who are not skilled, it would be very difficult. It was a risk, and the second one is if you print at the same height, even if it's totally well calibrated, if you print with one, the other one will cross it during the move because it's the same height.*

These challenges underscored the need for further refinements in both hardware design and firmware control to ensure reliable and user-friendly dual-extrusion printing.

The difficulties led PrintCo to redesign the dual-extrusion print head. Specifically, PrintCo revised both the electronics and mechanical design of the dual-extrusion print head. The outcome of this effort was a new print head design with liftable nozzles. This design resolved a key flaw of the experimental design, namely that the inactive nozzle bumped into the filament extruded by the other nozzle.

A final example for parametrizing components is a feature called active bed leveling. As PrintCo looked to meet user needs, they noticed that next to the temperature differences between the print head and the print bed, another key source of inaccurate prints was the poor calibration of the print bed. If the distance between the print bed and print head was too large or too small, the first layers of filament might not stick, which would lead to a problem colloquially referred to as spaghetti printing. The traditional way of calibrating the print bed was with calibration cards: Pieces of plastic in credit card format that were used to calibrate the distance between the print bed and print head as users manually adjusted the vertical position of the print bed. Manual calibration was a source of variation that negatively affected print outcomes.

In response, PrintCo explored ways to improve the current print bed calibration mechanism and designed an active bed leveling mechanism. This process began with the design of a novel capacitive sensor that was attached to the print head to measure the distance between the print head and the print bed. Doing so meant that PrintCo had to implement code for reading data from the new sensor, and this could be quite challenging, given a multitude of factors that could introduce noise into the measurement. A firmware engineer described these factors as follows:

> *Measurement [with this sensor] can be very accurate, but it depends on several conditions to work well. Blobs of material on the nozzle can affect the accuracy; while the nozzle is heated during measurements to minimize the impact of small amounts of filament, larger amounts will still interfere. Vibrations, such as those caused by placing the printer on a washing machine, can add significant noise to the sensor readings, so it's best not to use active leveling in such conditions. Additionally, ensure the fan cover is properly closed, as the sensor is attached to the fan bracket, and an improperly closed fan cover can cause problems. The sensor is also very sensitive to hands, so avoid holding your hands in the machine or touching the top of the print head, especially the screws, during the leveling process to prevent disruptions.*

These considerations highlighted the complexity of implementing active bed leveling and the necessity of refining the firmware to ensure reliable and accurate calibration under varying conditions.

Following this realization, PrintCo implemented several new machine codes to read data from the sensor for active bed leveling to determine the distance between the print bed and print head and adjust the vertical position of the print bed in response to that. Yet implementing these machine codes proved challenging, as the design of the firmware did not immediately parametrize the new component in the way PrintCo had anticipated. As a software architect told us:

> *The bed leveling part is interesting because there have been issues there for over a year, and people were really unsure of what the issues were because Mechanics thought, "Okay, we followed the concept very nicely." Electronics thought, "Okay, well, our sensor has these requirements." And software thought, "Well, our code, we've checked over it a hundred times. It should be fine." But still, there were many issues.*

These persistent challenges underscored the intricate dependencies between mechanical, electronic, and software components that necessitated further refinements of active bed leveling.

These issues led the firmware engineers to explore why these problems emerged and how they might be addressed—that is, what an optimal way of parametrizing the new sensor might be. A firmware engineer described this situation as follows:

> *Our new printer was about to be launched, so it was very critical, and they asked me to work on [the leveling]. And instead of looking at the codes, trying to fix any issue there, I started to just gather from all of the printers here, all the data from the sensors, and then I started to see, well, actually, the sensor, the way it's working, it's not as good as we were imagining. So, we really sat next to the printers to see it go wrong while being there. And then we saw, okay, actually the entire process, how it was thought of, you need to move your nozzle down, you take a measurement, was prone to errors. If there was residue [material in the nozzle], then everything goes wrong. So, the process itself was wrong. Some parameters needed to be different, we needed to heat up in a certain way.*

PrintCo subsequently adjusted the firmware to account for sensor noise and detect when measurements were faulty and required a sensor restart.

Taken together, parametrizing components was critical for layering the architecture of the printers. Introducing new functionality through digital technology meant that physical components first had to be parametrized to be addressable by digital code through a new adapter layer—firmware. Doing so enabled interactions between physical hardware and digital technology. By crafting both digital representations and machine code instructions, PrintCo advanced a digital firmware layer not as an immutable piece of software for controlling components but as an abstraction from the specific components constituting the product architecture and its primary functions so that they could be adapted to serve existing and emerging use cases. Ultimately, this allowed other layers to address the representations or machine code instructions to capture or operate both the static and dynamic aspects of physical components.

4.3 Arranging Digital Functionality

While the first technique made the printer's components addressable and controllable by digital code, our analysis also revealed a second technique that was crucial for layering the product architecture of PrintCo's printers: *Arranging digital functionality*, which coupled digital technology with an existing as well as emerging set of further digital components. This was necessary to establish information flows between newly created digital representations and machine code instructions pertaining to printer components and digital service layers higher in the architecture. Unlike parametrizing components, arranging digital functionality coupled digital components across functional layers to introduce further digital functionality.

Two distinct activities made up this technique at PrintCo. The first activity was the *design of functional stubs,* that is, providing specific patterns of possible interactions among multiple parameterized physical components to solve certain problems for users. Through the design of functional stubs, PrintCo created a new adapter layer within its product architecture, which we labeled the

configuration layer. The configuration layer sat atop the firmware layer and enabled further digital product innovation: Rather than leaving it to users to find out how to configure the printer and perform desired tasks, functional stubs helped calibrate the print process directly (e.g., by manipulating printer parameters through a GUI). The configuration layer consisted of configuration files—collections of printer-specific settings—that specified how a digital object should be sliced to adapt the printing process. By so doing, the same object could be printed in numerous different ways by adjusting printer parameters. Thus, the configuration layer connected the slicing software with the underlying firmware layer to access now parametrized physical components. The configuration layer was essential because parametrizing components alone was not enough to create value for users. As a product manager succinctly captured:

> *Sometimes what you see is people saying [a component] can do 10 things better … but that does not solve the problem for the customer. It must bring value to the customer, and that's often not only a technical solution. That is what comes out of it, but it is not what you solve. It took a lot of time to understand what kinds of problems we were solving. Then, to make this happen, we need certain technology.*

This realization emphasized that digital innovation was not just about expanding technical capabilities but about ensuring those capabilities translated into meaningful value for users.

The second activity involved in this technique was *rendering interfaces*. Rendering interfaces added an additional adapter layer to open the slicing tool (a higher-order service layer) to existing and future digital components in the form of functional software extensions or other digital services that could help to further enhance printer functionality. PrintCo believed that allowing users to customize their software would help address their local printing needs. As explained by a software architect:

> *We had the idea, you know, that it would be pretty cool if other people were able to make extensions that would come with [our slicing tool] or that they would be able to customize the tool to their needs themselves.*

This shift highlighted PrintCo's growing recognition that software extensions could significantly enhance the printer's adaptability and long-term value for users, and the goal was to enable extensions to draw on functional stubs. Together, the design of functional stubs and rendering interfaces ensured that various sets of digital components could interact with one another.

An example of the design of functional stubs is the introduction of several so-called "print modes," configurations that adapted the slicing process to account for technical bottlenecks of the printer, properties of the object users tried to print (e.g., geometrical properties), and designers' intents. These aspects could affect the performance of the printers if not accounted for. For instance, it was a known issue that some printers could have a slightly tilted print bed due to normal variation during the manufacturing process and that this tilt affected the dimensional accuracy of a print under certain conditions, as a firmware engineer mentioned:

> *[If the print bed] is skewed or bent—it's not exactly straight—then the layers aren't going to adhere and the printer's going to fail.*

This example illustrates how print modes were intended to address hardware limitations to ensure consistent print performance.

Similarly, objects with intricate details and complex geometrical shapes (e.g., overhang) were inherently challenging to print. Moreover, users, at times, have different objectives when printing, such as producing objects quickly or optimizing the visual quality of printed objects. Together, these factors constituted a large combinatorial space of possible configurations.

To ensure best possible print outcomes across the many configurations, PrintCo bundled settings into predefined configurations, print modes, that could be used to adapt and tune the slicing process. Initially, PrintCo introduced two different print modes, one that emphasized print speed and one that emphasized visual quality. When optimized for print speed, the printer operated with a higher filament extrusion rate and travel speed. Conversely, the visual quality print mode placed an emphasis on the outer layer and required a lower travel speed. Later, PrintCo expanded this adapter layer by introducing additional settings, such as infill patterns to improve part strength and new support structures for dual-extrusion printing. These print modes defined the various ways in which different printer components interacted with print objects, depending on user preferences. These patterns of interaction were the functional stubs that PrintCo made available in the slicing software through print modes to better meet user needs.

An example of rendering interfaces is how PrintCo enabled users to provide functional extensions—small plug-ins—to its slicing software by allowing them to interface with slicing tool functions. Many of PrintCo's users had a deep understanding of the printers' product architecture and often tweaked their printers to specific use cases. To better meet these users' unique requirements, PrintCo sought to provide them with additional options by making the implementation of functional extensions more attainable to this group of users.

To do so, PrintCo overhauled the architecture of the slicing tool with the aim of increasing the modularity of its components. While the slicing software had initially not been developed with that goal in mind, the overhaul ensured that individual components could be

implemented and added to without affecting the slicing tool's overall architecture. Specifically, PrintCo adopted an application development framework for building modular and extensible user interfaces, which made it easier for users to tailor the slicing tool to their needs through functional extensions. A senior software engineer described this situation as follows:

> *We made the software extremely plug-in-able, so pretty much anything in our slicing software is now a plug-in to make it tameable, all those kinds of things. All those things we designed with the idea that it would be easier for others to contribute.* (Senior software engineer)

The framework provided access to slicing tool functions as discrete components that could be independently modified or replaced. This modular design enabled users to add custom panels, modify settings dialogs, or integrate new features without altering the core code. This framework provided an interface layer to the software. Thus, users could add functional extensions to the printer architecture. For instance, a user developed a functional extension for generating custom support structures for dual-extrusion printing. Another example is a functional extension for unit conversion, from metric to imperial for international users. In all, this technique yielded adapter layers that helped couple layers of digital components with one another.

4.4 Interactions Among the Techniques

The two techniques—parametrizing physical components and arranging digital functionality—explain how PrintCo layered the product architecture of its 3D printers, which initially consisted of largely decoupled digital and physical elements, by embedding digital technology components within it. At the core of these techniques were adapter layers that loosely coupled digital components (in particular, the slicing software) with the product architecture and created further opportunities for digital functionality to be introduced. Importantly, both techniques were logically and temporally connected. Before PrintCo could enable extensions to its slicing software, it needed to parametrize physical components such that they became digitally controllable, configurable, and adaptable. This means that layering began "at the bottom" of the product architecture, with the process of parametrizing the core physical machine components of the printer, which PrintCo first digitally parametrized to then create functional stubs on top of parametrization that bundled core physical operations (such as turn and move operations) into abstract printing routines (e.g., print a circle).

Finalizing the design of its functional stubs allowed PrintCo to logically "move up" the product architecture to create an adapter layer that connected the slicing software and printer hardware. Only then was PrintCo in a position to render interfaces that would open the slicing software's GUI to functional extensions. In this sense, arranging digital functionality enhanced the layering of the product architecture by iteratively adding three additional layers to the product architecture from the bottom up, two below and one atop the slicing software. These new *adapter layers* loosely coupled parametrized hardware with the slicing software through dedicated firmware and configuration layers: one (firmware) resided in the physical component layer of the product, while the other resided within a digital layer only. Adding a new interface layer on top of the slicing software that constituted the layer of main digital services, in turn, opened up the product for additional functionality that could now be loosely coupled with the slicing software and the functional stubs created to operate the parametrized physical components in the printer.

5 Discussion

Our study identifies two techniques that organizations use to embed digital components within a product architecture. Figure 3 represents our findings conceptually, in the spirit of Lyytinen's (2022) visualization. The model in Figure 3 suggests that layering a digital product architecture is an iterative bottom-up process[3] through which organizations couple digital and physical components through the design of adapter layers. Adapter layers establish sets of digital and physical components as functional layers within the architecture. Recall how PrintCo initially developed its slicing software and printers independently but then realized the potential to improve the performance of its printers by adapting the slicing software. To do so, PrintCo parametrized its printers' physical components to be able to loosely couple the slicing software with the printer. PrintCo accomplished this progress by implementing dedicated layers that help with the coupling and flexible operation of (already existing or newly introduced) layers of functional components.

The model in Figure 3 also suggests that the scope of layering expands vertically over time. Initially, layering occurs at the lower end of the product—at the level of physical components that must be parametrized to become accessible to digital components. PrintCo created a firmware layer to represent and control its physical components in digital form. Later, layering logically "moves upward" from the physical component layers to eventually cover the entire architecture. PrintCo, for

3 In the organizational literature, bottom-up process are usually understood as unplanned patterns of action in the day-to-day practices of employees that shape strategic objectives and action (Mintzberg & Waters, 1985); in this paper, we simply mean activities in the design or change of a product architecture that are at the lower, physical end of products that involve higher-order functionality on different layers of the architecture.

example, leveraged the parametrized physical components in the design of additional digital functionality "on top" of the product. At the higher layers of the product architecture, adapter layers are created to arrange functionality (i.e., component interactions) such that additional software layers can be added to create value for users. Adapter layers are key because they can be used to dynamically couple various types of digital components (e.g., firmware, application software, complementary apps) with each other. Thus, organizations create adapter layers within product architectures to arrange the functionality of their different components into distinct and vertically stacked layers. These findings have several theoretical implications, which we discuss in turn.

5.1 Separating Functional and Adapter Layers in Layered-Modular Architectures

Our findings contribute to our understanding of layered-modular architectures, often regarded as the key organizing principle of digital innovation (Hylving & Schultze, 2020; Yoo et al., 2010). Specifically, we unpacked the techniques through which layers are implemented. For some digital product innovations, physical components primarily come in the form of general-purpose computing hardware that can readily be embedded with digital technology components for creating additional value. In this setting, the hardware is primarily a rigid vessel (cf. Goebeler et al., 2024) waiting to be enhanced with digital technology. And while this may be the case for some digital product innovations (e.g., Svahn et al., 2017), our study shows that this is not the case for products with more specialized mechatronic physical components, like the digitalized movie theatre discussed in Wang et al. (2022) or, indeed, the 3D printing machines we studied. While PrintCo was eager to enhance its printers through its slicing software, doing so required interactions among physical components to be more flexible, and any improvements in the slicing software needed to be coupled with printer components through functional stubs.

Our findings suggest that organizations design adapter layers to facilitate connections between and across the layers that make up the final product architecture. Adapters allow different functional layers to communicate with one another and ultimately enable the joint specialization of those layers while maintaining the possibility of independently developing individual, potentially modular components within a layer.

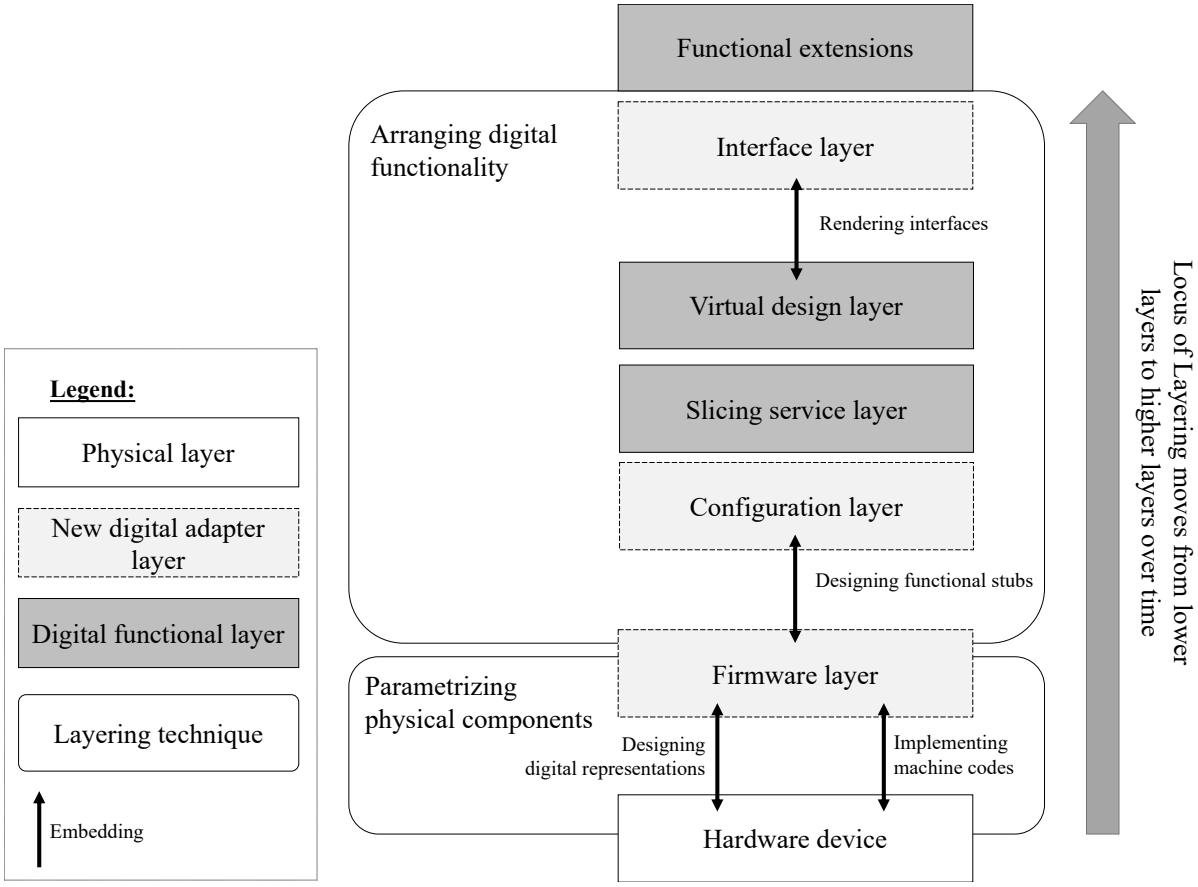

Figure 3. Conceptual Model of Layering Digital Technology Components Into a Physical Product Architecture

Taken together, these findings suggest that within the layered-modular architecture, not all layers are equal: While some layers are functional, that is, the dominant source of innovation, as described in prior work (Yoo et al., 2010), adapter layers are important because they enable and control information flows across functional layers. This insight contrasts with research suggesting that innovation can happen on any layer, with minimal consideration of other layers (Yoo, 2013; Yoo et al., 2012). Our analysis suggests that adapter layers are a key enabler for innovation on functional layers. Imagine a new navigation app for a smartphone that is useless without a properly parametrized and accessible GPS sensor.

Adapter layers bear comparison with interfaces (e.g., Pujadas et al., 2024) and boundary resources (e.g., Eaton et al., 2015), which have both been shown to play a pivotal role in digital innovation. Interfaces define how individual components must be designed to be interoperable with one another (Baldwin & Clark, 2000; Simon, 1996). Boundary resources wrap around interfaces and provide additional tools that enable an arm's length relationship with developers who contribute to a product (Eaton et al., 2015). In the end, interfaces and boundary resources ensure compatibility between components, both internal and external. Adapter layers are similar to interfaces because they ensure addressability and compatibility across functional layers. However, while interfaces define the rules, methods, and properties that components must implement to interact consistently, they typically provide no functionality but only *access to* functionality (Ghazawneh & Henfridsson, 2013). For instance, Baldwin and Clark (2000) describe interfaces in terms of design rules as purely declarative contracts. Adapters, by contrast, are implementation-level solutions that enable communication among otherwise unrelated components by translating or converting one interface into another, allowing them to work together without changing their underlying structure. This enables adapter layers to operate across and integrate multiple components (Lee & Berente, 2012) and the layer as a whole, rather than individual components. Thus, adapter layers enable innovation in functional layers, because they have the capacity for change, while interfaces typically remain stable (Baldwin & Clark, 2000). Adapter layers enable interactions among functional layers, and changes in functional layers can create value for users of a digital product innovation. In this light, adapter layers are key to the open-ended recombination across the layered-modular architectures of various digital products (Henfridsson et al., 2018).

Our study also highlights the crucial role of firmware in digital product innovation. Firmware has, to date, not been considered a source of innovation, but is rather viewed as a static means for controlling physical components (Hylving & Schultze, 2020; Lee & Berente, 2012). The locus of digital innovation so far in the literature has been squarely rooted in the digital technology components on content or service layers (Nambisan et al., 2017; Yoo et al., 2010). In contrast, our analysis uncovered two fundamental activities, designing digital representations and implementing machine codes, that are key to embedding a firmware layer within the product architecture.

This finding carries two significant implications. On the one hand, it suggests that the scope of digital innovation research must extend beyond the realm of digital technology application and service layers to encompass a more expansive set of digital technology components that includes not only apps, data, and algorithms but also firmware. The key distinction of firmware as a digital technology is its inscription into physical devices. It does not stand as decoupled and ephemeral digital material (von Briel et al., 2018); rather, it is, by design, embedded in and constrained by the geospatial attributes of the physical components on which it rests, such as their size, place, material composition, or even weight. Focusing on firmware thus provides several opportunities to expand upon our insights. For example, our study is centered on the early stages of digital product innovation, where firmware is key to embedding digital components into product architectures that make physical components addressable through digital components. However, we have not yet examined how the role of firmware may change in the later stages of digital product innovation. It is conceivable that firmware may be more malleable and not as firm as previously thought.

Finally, our findings suggest an extension to the layered-modular architecture. In Yoo et al.'s (2010) seminal work on digital innovation, firmware—subsumed under "logical capability"—plays a subordinate role rather than serving as a source of innovation. Our work echoes recent research that has suggested that the architecture of digital product innovations can be more accurately described as layered hierarchical rather than modular (Hylving & Schultze, 2020). It extends this line of inquiry by theorizing the specific techniques that layer product architectures (Henfridsson et al., 2018; Yoo, 2010; Yoo et al., 2012). That is, by parametrizing components and arranging functionality, organizations install adapter layers into product architectures that can couple physical to digital and digital to other digital components to introduce new functionality (Holmström, 2018; Hylving & Schultze, 2020; Sandberg et al., 2020). This suggests that a digital product innovation's logical capability—firmware—may play a more central role in enabling combinatorial innovation processes than assumed by the available literature because it sits right at the intersection between physical and digital components.

5.2 The Locus of Layering: Ascending Architectural Layers

Our study also provides insights into how layering unfolds over time. The digital innovation literature suggests that organizations achieve a layered-modular architecture by adding layers of digital technology "on top of" existing product architectures (e.g., Sandberg et al., 2020; Yoo et al., 2012), and that such design moves can occur at any time and without a fixed sequence (Henfridsson et al., 2018). But doing so requires that products are already amenable to being embedded with digital technology and flexible enough to cater to emerging and perhaps unanticipated use cases (Zittrain, 2006). If this is not the case, any digital technology added to a product architecture may be of limited value to users because components remain decoupled. Our study reveals that rather than beginning with high-level software, layering begins in the lower product layers, involving only isolated sets of components before becoming increasingly expansive and ascending the product architecture.

This finding suggests that path dependencies exist in layering that may be more pronounced than previously thought (Henfridsson et al., 2018). Organizations need to attend to these interdependencies to be able to introduce new digital functionality on higher layers to extend a product architecture's scope. For instance, PrintCo iterated between the hardware and firmware layers in the design of digital representations and machine code instructions. Thus, layering begins in physical components and ascends architectural layers over time. Key to this evolution is enabling a larger set of interactions among product components that digital technology components can draw on. Recall how PrintCo used collections of print settings to adapt the printing process to meet users' preferences more effectively: Only after PrintCo wrapped its printers in a layer of firmware could PrintCo explore how to co-specialize its slicing software and incorporate external innovation. As such, although layering involves considerable iterative development within individual layers, the locus of layering ascents along an axis of abstraction in the product architecture.

The insight that layering begins on lower layers before ascending to higher layers extends our knowledge about how organizations can meaningfully add digital technology to their market offerings (e.g., Sandberg et al., 2020). This work suggests that adding digital technology enables new interactions among an emergent set of actors, thus rendering products gradually more generative (Fürstenau et al., 2023). Our study shows that new interactions, eventually unfolding across multiple layers, initially begin from a narrow focus on the lower layers. The insight also mirrors the view of Hylving and Schultze (2020) that digital product innovation resembles a layered hierarchical rather than strictly modular architecture, a distinction also noted recently in other studies (e.g., Lorenz et al., 2024).

5.3 Operationalizing Layering

Finally, our study also contributes to the literature on technology and innovation management (Baldwin, 2023; Galunic & Eisenhardt, 2001; Henderson & Clark, 1990). The techniques PrintCo used to embed digital technology within its product architecture complement knowledge about "design rationalization processes" that drive the modularization of product architectures (Baldwin & Clark, 2000).[4] Parametrizing components and arranging functionality have in common with design rationalization that they hide and encapsulate some parts of a product and abstract details from their implementation to manage complexity. At the same time, the techniques we describe also differ in the objectives they pursue from the design rationalization processes discussed in the literature on modularity (Baldwin & Clark, 2000). Design rationalization processes, such as substituting or augmenting, emphasize independence among modules: they allow modules to be developed, maintained, and replaced without affecting the rest of the system.

In contrast, the layering techniques we found focus on how complementary components—hardware and software—can be brought together in the first place, namely by creating dedicated adapter layers that facilitate communication among previously separate components. These points of contrast and comparison between layering and modularization imply that both are amenable to deliberate operationalization: Just like modular operators, layering, as we describe it, is actionable for innovators.

5.4 Limitations

Our study has several limitations that present opportunities for further research. First, our case selection forms a boundary condition for generalizing our findings. 3D printers are a specific kind of digital product innovation, which vary in the extent to which physical or digital components dominate behavior, functionality, meaning, or value (von Briel et al., 2018; Wang, 2021). Understanding in depth how parametrizing components and arranging functionality are required for and enable recombination in different digital product innovations such as, for example, consumer devices (e.g., smart home or wearable technology) versus industrial grade products (e.g., 3D printers or autonomous cars) or even large-scale systems (e.g., digital theatres) presents a stimulating

[4] For example, substituting is a design rationalization process in which organizations replace one product component with another, higher-performing, component. Augmenting is another design rationalization process, in which organizations add further modules to a product architecture to enhance its functionality (Baldwin & Clark, 2000).

research opportunity to explore the robustness, boundary conditions, and possibilities for analytical generalization of the two techniques we developed and of the role of firmware in digital product innovation more generally.

Further, we explicitly focused on the "artifact" that is the outcome of digital innovation, namely the 3D printer, its product components, and architecture. We deliberately excluded questions of organizing (Lee & Berente, 2012; Yoo et al., 2012), innovation tools (Marion & Fixson, 2021; Zhang et al., 2021), or exogenous influences such as financing or infrastructure that might conceivably shape the design of digital product innovations. For example, PrintCo raised several rounds of venture capital, grew substantially, and experienced high employee attrition—all of which could have conceivably affected the way they engaged in digital innovation as a process.

Finally, as in other inductive qualitative field studies, there is inherent subjectivity in our analysis and interpretation of the collected data. Our primary source of data was interviews, some of which were retrospective. This strategy is prone to interviewee bias, recency bias, and selection bias, which could have impacted the accuracy of the reported data. By using a variety of data sources (e.g., both company and public documents) and focusing on key events that are publicly traceable, we tried to mitigate these biases. Furthermore, our analysis and findings were influenced by our use of the literature on digital innovation and product architecture as sensitizing lenses and the way in which we conducted open and axial coding. To ensure rigor in our procedures, we followed the typical iterative process of analyzing data, engaging with literature, and collecting new data (Charmaz, 2006; Glaser & Strauss, 1967; Urquhart et al., 2010). We used theoretical sampling logic to identify follow-up data collection (both documents and interviewees) to query specific outcomes of our iterative analysis outcomes (e.g., identifying main innovation episodes, key events, or specific aspects of firmware design). In our team, we constantly challenged each other's interpretations and conclusions and also engaged with case informants to test our emerging explanations. Finally, we developed our emerging theoretical model by drawing on guidelines for scaling up from data to concepts to categories (Urquhart et al., 2010).

6 Conclusion

Organizations still struggle to consistently develop and deliver successful digital product innovations because digital technology cannot merely be tacked on to product architectures to create value and unlock generative potential. We show that the successful design of digital product innovations requires organizations to embed adapter layers into product architectures, so that digital and physical components in a product architecture are not only modularized but also layered to enable access, connections, and ultimately recombination.

Acknowledgments

We are thankful to Dorothy E. Leidner for her guidance and support throughout the review process. We also appreciate the anonymous reviewers, whose careful and constructive comments helped us to sharpen our arguments and refine our contributions. This work was supported by a grant from the German Research Foundation (DFG, grant ID 453416399).

References

Abbott, A. (1990). A primer on sequence methods. *Organization Science*, *1*(4), 375-392.

Alaimo, C., & Kallinikos, J. (2022). Organizations decentered: data objects, technology and knowledge. *Organization Science*, *33*(1), 19-37.

Albert, D., & Siggelkow, N. (2022). Architectural search and innovation. *Organization Science*, *33*(1), 275-292.

Baldwin, C. Y. (2023). Design rules: Past and future. *Industrial and Corporate Change*, *32*(1), 11-27.

Baldwin, C. Y., & Clark, K. B. (2000). *Design rules, Volume 1: The power of modularity*. MIT Press.

Baskerville, R., Myers, M. D., & Yoo, Y. (2020). Digital first: The ontological reversal and new challenges for IS research. *MIS Quarterly*, *44*(2), 509-523.

Berends, H., & Deken, F. (2021). Composing qualitative process research. *Strategic Organization*, *19*(1), 134-146.

Brusoni, S., & Prencipe, A. (2001). Unpacking the black box of modularity: technologies, products and organizations. *Industrial and Corporate Change*, *10*(1), 179-205.

Charmaz, K. C. (2006). *Constructing grounded theory: A practical guide through qualitative analysis*. SAGE.

Colfer, L. J., & Baldwin, C. Y. (2016). The mirroring hypothesis: Theory, evidence, and exceptions. *Industrial and Corporate Change*, *25*(5), 709-738.

Eaton, B., Elaluf-Calderwood, S., Sørensen, C., & Yoo, Y. (2015). Distributed tuning of boundary resources: the case of Apple's iOS service system. *MIS Quarterly*, *39*(1), 217-243.

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

Fürstenau, D., Baiyere, A., Schewina, K., Schulte-Althoff, M., & Rothe, H. (2023). Extended generativity theory on digital platforms. *Information Systems Research*, *34*(4), 1686–1710.

Galunic, D. C., & Eisenhardt, K. M. (2001). Architectural innovation and modular corporate forms. *Academy of Management Journal*, *44*(6), 1229-1249.

Garud, R., Jain, S., & Tuertscher, P. (2008). Incomplete by design and designing for incompleteness. *Organization Studies*, *29*(3), 351-371.

Gawer, A. (2021). Digital platforms' boundaries: The interplay of firm scope, platform sides, and digital interfaces. *Long Range Planning*, *54*(5), Article 102045.

Ghazawneh, A., & Henfridsson, O. (2013). Balancing platform control and external contribution in third-party development: the boundary resources model. *Information Systems Journal*, *23*(2), 173–192.

Gioia, D. A., Corley, K. G., & Hamilton, A. L. (2013). Seeking qualitative rigor in inductive research. *Organizational Research Methods*, *16*(1), 15-31.

Giustiziero, G., Kretschmer, T., Somaya, D., & Wu, B. (2023). Hyperspecialization and hyperscaling: A resource-based theory of the digital firm. *Strategic Management Journal*, *44*(6), 1391-1424.

Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research*. Aldine.

Goebeler, L., Hukal, P., & Xiao, X. (2024). Four roles of physicality in digital innovation: A theoretical review. *Journal of Strategic Information Systems*, *33*(4), Article 101862.

Henderson, R. M., & Clark, K. B. (1990). Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*, *35*(1), 9-30.

Henfridsson, O., Mathiassen, L., & Svahn, F. (2014). Managing technological change in the digital age: The role of architectural frames. *Journal of Information Technology*, *29*(1), 27-43.

Henfridsson, O., Nandhakumar, J., Scarbrough, H., & Panourgias, N. (2018). Recombination in the open-ended value landscape of digital innovation. *Information and Organization*, *28*(2), 89-100.

Holmström, J. (2018). Recombination in digital innovation: challenges, opportunities, and the importance of a theoretical framework. *Information and Organization*, *28*(2), 107-110.

Huang, J., Henfridsson, O., & Liu, M. J. (2022). Extending digital ventures through templating. *Information Systems Research*, *33*(1), 285-310.

Hylving, L., & Schultze, U. (2020). Accomplishing the layered modular architecture in digital innovation: The case of the car's driver information module. *Journal of Strategic Information Systems*, *29*(3), 101621.

Kuan, J., & West, J. (2023). Interfaces, modularity and ecosystem emergence: How DARPA modularized the semiconductor ecosystem. *Research Policy*, *52*(8), 104789.

Langley, A. (1999). Strategies for theorizing from process data. *Academy of Management Review*, *24*(4), 691-711.

Lee, J., & Berente, N. (2012). Digital innovation and the division of innovative labor: Digital controls in the automotive industry. *Organization Science*, *23*(5), 1428-1447.

Lehmann, J., Recker, J., Yoo, Y., & Rosenkranz, C. (2022). Designing digital market offerings: How digital ventures navigate the tension between generative digital technology and the existing environment.

MIS Quarterly, *46*(3), 1453-1482.

Lorenz, J., Chandra Kruse, L., & Recker, J. (2024). Creating and capturing value with physical-digital experiential consumer offerings. *Journal of Management Information Systems*, *41*(3), 779-811.

Lyytinen, K. (2022). Innovation logics in the digital era: A Systemic review of the emerging digital innovation regime. *Innovation: Organization & Management*, *24*(1), 13-34.

Lyytinen, K., Yoo, Y., & Boland, R. J. (2016). Digital product innovation within four classes of innovation networks. *Information Systems Journal*, *26*(1), 47-75.

MacCormack, A., Rusnak, J., & Baldwin, C. Y. (2006). Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, *52*(7), 1015-1030.

Marion, T. J., & Fixson, S. K. (2021). The transformation of the innovation process: How digital tools are changing work, collaboration, and organizations in new product development. *Journal of Product Innovation Management*, *38*(1), 192-215.

Mintzberg, H., & Waters, J. A. (1985). Of strategies, deliberate and emergent. *Strategic Management Journal*, *6*(3), 257-272.

Nambisan, S., Lyytinen, K., Majchrzak, A., & Song, M. (2017). Digital innovation management: reinventing innovation management research in a digital world. *MIS Quarterly*, *41*(1), 223-238.

Parker, G., & Van Alstyne, M. (2018). Innovation, openness, and platform control. *Management Science*, *64*(7), 3015-3032.

Parker, G., Van Alstyne, M., & Jiang, X. (2017). Platform ecosystems: How developers invert the firm. *MIS Quarterly*, *41*(1), 255-266.

Pujadas, R., Valderrama, E., & Venters, W. (2024). The value and structuring role of web APIs in digital innovation ecosystems: The case of the online travel ecosystem. *Research Policy*, *53*(2), 104931.

Rayna, T., & West, J. (2023). Where digital meets physical innovation: Reverse salients and the unrealized dreams of 3D printing. *Journal of Product Innovation Management*, *40*(4), 530-553.

Sandberg, J., Holmström, J., & Lyytinen, K. (2020). Digitization and phase transitions in platform organizing logics: Evidence from the process automation industry. *MIS Quarterly*, *44*(1), 129-153.

Sarker, S., Xiao, X., Beaulieu, T., & Lee, A. S. (2018). Learning from first-generation qualitative approaches in the IS discipline: An evolutionary view and some implications for authors and evaluators (PART 1/2). *Journal of the Association for Information Systems*, *19*(8), 752-774.

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). MIT Press.

Strauss, A. L., & Corbin, J. (1998). *Basics of qualitative research: techniques and procedures for developing grounded theory* (2nd ed.). SAGE.

Svahn, F., Mathiassen, L., & Lindgren, R. (2017). Embracing digital innovation in incumbent firms: How Volvo Cars managed competing concerns. *MIS Quarterly*, *41*(1), 239-253.

Tilson, D., Lyytinen, K., & Sørensen, C. (2010). Digital infrastructures: The missing IS research agenda. *Information Systems Research*, *21*(4), 748-459.

Urquhart, C., Lehmann, H., & Myers, M. D. (2010). Putting the theory back into grounded theory: Guidelines for grounded theory studies in information systems. *Information Systems Journal*, *20*(4), 357-381.

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

Wang, G., Henfridsson, O., Nandhakumar, J., & Yoo, Y. (2022). Product meaning in digital product innovation. *MIS Quarterly*, *46*(2), 947-976.

Wang, P. (2021). Connecting the parts with the whole: Toward an information ecology theory of digital innovation ecosystems. *MIS Quarterly*, *45*(1), 397-422.

West, J., & Kuk, G. (2016). The complementarity of openness: How MakerBot leveraged Thingiverse in 3D printing. *Technological Forecasting and Social Change*, *102*, 169-181.

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

Yoo, Y. (2013). The tables have turned: How can the information systems field contribute to technology and innovation management research? *Journal of the Association for Information Systems*, *14*(5), 227-236.

Yoo, Y., Boland, R. J., Lyytinen, K., & Majchrzak, A. (2012). Organizing for innovation in the digitized world. *Organization Science*, *23*(5), 1398-1408.

Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). The new organizing logic of digital innovation: An agenda for information systems research. *Information Systems Research*, *21*(4), 724-735.

Zhang, Z., Lindberg, A., Lyytinen, K., & Yoo, Y. (2021). The unknowability of autonomous tools and the liminal experience of their use. *Information Systems Research*, *32*(4), 1192-1213.

Zittrain, J. L. (2006). The generative internet. *Harvard Law Review*, *119*, 1974-2040.

About the Authors

Julian Lehmann is an assistant professor of information systems at Arizona State University, Julian studies how firms create strategic value from digital technology in both startup and incumbent firms. His work draws primarily on qualitative process research methods such as case study research. Julian was a member of the AIS SIG DITE executive leadership from 2022 to 2024. He is also the organizer and chair of the annual SIG DITE Paper Development Workshop. He obtained a Ph.D. in information systems from the University of Cologne.

Philipp Hukal is an associate professor of innovation in the Department of Strategy and Entrepreneurship at BI Norwegian Business School in Oslo, Norway. His research focuses on digital innovation within and across organizations, covering topics such as new ventures, digital platforms, and organizational transformation. He holds a PhD in information systems and management from Warwick Business School – The University of Warwick (UK).

Jan Recker is Nucleus Professor for Information Systems and Digital Innovation at the University of Hamburg Business School, Germany, and an adjunct professor at the QUT Business School, Australia. He received his PhD from Queensland University of Technology. His research focuses on digital innovation and entrepreneurship, digital solutions for sustainable development, and systems analysis and design. Many people like his *This IS Research* podcast more than his papers.

Sanja Tumbas is an innovation catalyst at the Institute for Digital Technology Management at the Bern University of Applied Sciences in Switzerland. She holds a PhD from the University of Liechtenstein. Her work focuses on bridging research teams with industry and society to foster collaboration and facilitate knowledge transfer. Her fields of interest include digital and deep-tech entrepreneurship, science-based ventures, and the interplay between software-centric and hardware-centric innovation strategies.