applied sciences

MDPI

Article

Evaluating the Effectiveness of Designs for Low-Cost Digital Manufacturing Systems

Jan Kaiser *, Gregory Hawkridge, Anandarup Mukherjee and Duncan McFarlane

Department of Engineering, Institute for Manufacturing, University of Cambridge, 17 Charles Babbage Road, Cambridge CB3 0FS, UK
* Correspondence: jk823@cam.ac.uk

Abstract: There are many well-known systematic approaches to design the digital systems used in manufacturing. However, there are only a few approaches that specifically deal with *low-cost* components. Such components may not provide the same level of completeness as more expensive industrial alternatives and may need to be combined with other components to become comparable. Consequently, common design challenges for systems comprising such low-cost components revolve around extendability and interface standardisation. There is a need for analysing the capability of the existing approaches to design these systems. This study aims to evaluate the effectiveness of designs for low-cost digital manufacturing systems that have been derived from a particular design approach. The proposed evaluation methodology is used for the special case of designs that are directly based on reference architectures and for the development of specific metrics for that purpose. To quantify the effectiveness, these metrics are applied to a number of design scenarios. Although focusing on reference-architecture-based designs, the proposed methodology can also be used for other design approaches. The evaluation and structured implementation comparison helps practitioners in selecting an effective design approach to low-cost digital manufacturing systems and provides insights into how a particular design approach can become more effective.

Keywords: design cost; reference architectures; metrics; low cost; digital manufacturing; design structure matrix; Industry 4.0

Citation: Kaiser, J.; Hawkridge, G.; Mukherjee, A.; McFarlane, D. Evaluating the Effectiveness of Designs for Low-Cost Digital Manufacturing Systems. *Appl. Sci.* **2023**, *13*, 12618. https://doi.org/10.3390/app132312618

Academic Editors: Benjamim Fonseca, Ivo Pereira and Ana Madureira

Received: 5 October 2023
Revised: 17 November 2023
Accepted: 21 November 2023
Published: 23 November 2023

1. Introduction

Design approaches describe the systematic creation of systems by analysing system requirements and using appropriate design techniques [1]. There are many well-know systematic design approaches to digital systems used in manufacturing [2–6]. They cover a wide range of industrial applications, from an individual machine monitoring system to a fully integrated factory. However, there are only a limited number of systematic design approaches that specifically deal with low-cost components. Most low-cost digital system deployments have been designed in an ad hoc manner and have only emphasised specific aspects of the development process. In particular, designers are mainly concerned with reducing the procurement costs of hardware and software components [7–10]. Low-cost components may not offer the same level of completeness as more expensive industrial alternatives and may need to be combined with other components to become comparable. Hence, common design challenges for systems comprising such low-cost components revolve around extendability and interface standardisation. These design challenges are particularly relevant to small and medium-sized enterprises (SMEs). In contrast to larger companies, SMEs lack the necessary technical skills and resources that are required to adopt digital technologies and build these low-cost digital manufacturing systems [11–13]. There is a need for analysing the capability of current design approaches to effectively support the design of low-cost digital manufacturing systems.

Digital manufacturing revolves around the application of digital information for the enhancement of manufacturing processes, supply chains, products and services [14]. Sev-

eral concepts fall under the umbrella of 'digital manufacturing', such as cyber-physical systems (CPS) and Internet of Things (IoT). Above all, the design of digital manufacturing systems is particularly relevant for Industry 4.0. By combining CPS, IoT and other disruptive decentralising technologies, Industry 4.0 aims to achieve a horizontal integration of manufacturing and enterprise systems [15]. Effectively designing such decentralised digital systems to enhance production and the control of manufacturing resources is a key concern of Industry 4.0.

In this study, we focus on the systematic design of particularly *low-cost* digital systems used in manufacturing. *Low-cost digital manufacturing* refers to *the development of digital systems to meet specific operational needs and for which the overall cost of deployment it keeps low* [16]. A *low-cost digital manufacturing system* is a special type of digital system that consists of a combination of affordable hardware and software components and satisfies the entry-level digitalisation needs of a manufacturing SME [16]. Low-cost digital systems can be used to deliver a *solution* to a particular problem. For example, these systems can be used to digitise a core production process or help integrate a legacy system into an IT environment [12].

The low-cost digital systems considered here characterise systems that are typically peripheral to core production processes, such as tracking and monitoring applications [13]. It is crucial to lay a foundation based on these elementary low-cost systems to help implement and adapt more complex integrated digital systems [17], such as a digital twin mirroring a factory that consists of various connected systems and applications [18,19]. A key challenge when building low-cost digital systems is that the cost is not only governed by the component costs alone but also comprises the time spent during the design, development and deployment [20]. As part of these 'design costs', this study evaluates the effectiveness of designs for low-cost digital manufacturing systems in terms of the effort required to design these systems. We define *design effort* as *the combination of physical, virtual and human resources required to develop, implement and deploy a digital system*. For example, physical and virtual resources represent hardware components and the data of a digital system, whereas human resources include the time spent during development.

This study is concerned with effectively designing low-cost digital systems used in manufacturing. Specifically, the aim of this study is to evaluate the effectiveness of designs for low-cost digital manufacturing systems that have been derived from a particular design approach. The proposed evaluation methodology is used for the *special case of designs that are directly based on reference architectures* and for the development of specific metrics for that purpose. To quantify the effectiveness, these metrics are then applied to a number of design scenarios relevant to low-cost digital systems. While in this study we concentrate on reference-architecture-based designs, the proposed methodology can also be used for other design approaches, such as object-oriented programming. The evaluation and structured implementation comparison helps practitioners in selecting an effective approach to design low-cost digital manufacturing systems and provides insights into how a particular design approach can become more effective. Existing approaches for quantitatively evaluating architectural designs only focus on system architectures that are dedicated to a specific application. While the proposed evaluation metrics affect future system designs, more work is required to develop a model capable of making proper design predictions. The key research gaps addressed in this study include the need for a quantitative evaluation of designs based on a reference architecture, and the need for an effectiveness assessment of low-cost digital system designs.

The paper is structured as follows. Section 2 provides an overview of design approaches to digital manufacturing, commonly used evaluation metrics for digital systems, the challenges developers face when designing low-cost digital systems and approaches to evaluate designs based on an architecture. Section 3 proposes the evaluation methodology of this study. The next two sections constitute the results of the work. Section 4 develops the effectiveness measures and metrics and implements the evaluation testbed, while Section 5 performs the quantitative effectiveness evaluation by applying the measures and metrics to the testbed. Section 6 discusses the results of this study before relevant conclusions are drawn.

2. Background

Design approaches support the systematic creation of systems through an analysis of system requirements and the application of appropriate design techniques. Much work has been conducted on structuring the design of digital systems in manufacturing. However, one limitation of these studies to date has been the *lack of a systematic approach to design low-cost digital systems*. This study evaluates the effectiveness of designs for low-cost digital manufacturing systems for the special case of designs that are directly based on reference architectures. This section describes the research gaps in the study of designing low-cost digital manufacturing systems. We begin by reviewing general design approaches to digital manufacturing, which is followed by an overview of measures and metrics commonly used for evaluating digital system designs. Then, the key challenges developers face when designing low-cost digital systems are described before previous approaches for evaluating architectural designs are reviewed.

2.1. Design Approaches to Digital Manufacturing Systems

Digital manufacturing systems can be designed in various ways, and not all of them are mutually exclusive. For example, a system-oriented view on design is provided by Moses [3], who identifies three general *design methodologies* for engineering systems: First, the *top-down tree-structured* methodology divides an engineering problem into subproblems that can either be solved or are further divided until they can be solved. When representing the design process as a graph, subproblems constitute the nodes, whereas the development path defines a sequence of edges connecting a sequence of nodes. Each node has exactly one parent node, which remains fixed. Second, the *layered* or *platform-based* methodology describes a hierarchical design process where elements of a layer can be viewed as part of a group. Parent nodes can readily change which leads to more development paths from top to bottom. For the *network-based* methodology, systems are designed as a decentralised network of connected components.

Another design approach describes the use of *architectures* for the design of digital systems. Architectures are models that abstract the structure and function of a system [4]. There are different types of architectures, including reference architectures, system architectures, platforms and frameworks, which vary in their level of abstraction and technologies used (e.g., service-orientation) [14]. While there is generally no explicit connection to one of the aforementioned design methodologies, most architectures for digital systems adopt a layered or network-based design methodology.

To address the requirements of large complex systems, Farid and Suh [2] formalise the design process via the *axiomatic design* paradigm, which maps functional requirements to the physical architecture of a system. Two design axioms are used for guidance, namely the independence axiom and the information axiom. While the former requires the set of functional system requirements to be mutually exclusive and collectively exhausted, the latter aims to minimise the information content of a system design.

In software engineering, *object orientation* describes a structured way of programming, in which objects encapsulate inherent processing abilities and message communication and ensure uniformity of appearance, status and reference [5]. For example, data structures, variables or methods characterise objects. This concept can be expanded to enable inheritance, where one object is derived from another object. Abstracting an object-oriented solution to solve more general design problems results in a design pattern that is similar to an architectural design approach [6].

2.2. Measures and Metrics for Evaluating Digital System Designs

There are a number of well-established measures and metrics to evaluate digital system designs, which form the basis for the quantitative effectiveness evaluation conducted in this study. *Metrics* are observable values that are directly measured and can be collected automatically [21]. For software systems, metrics produce numbers that describe properties of source code [22]. These metrics can be expanded to develop prediction models for resource

requirements and software quality. In contrast, *measures* associate meaning with these observable values by deriving interpretations from one or more metrics. While measures can only be used for a qualitative assessment, several metrics can be deduced from them.

The first attempts towards creating metrics for evaluating digital systems goes back to the 1960s. Developers introduced the *lines of code (LOCs)* metric to measure the programming productivity and effort of software [22]. However, solely analysing the LOCs metric is insufficient to evaluate the complexity and cost of a system [2,23]. The complexity of a software system can be assessed, for instance, by using the cyclomatic complexity, which measures the number of linearly independent paths through a program [24]. Alternatively, Stevens et al. [25] introduced the concepts of coupling and cohesion in system design. Coupling measures the strength of association of a connection *between* modules. A strong coupling means that modules are highly interrelated. Cohesion gauges the degree of binding of elements *within* a module. A highly cohesive module reduces its coupling. For managing information of engineering applications, the *Design Structure Matrix (DSM)* has been introduced by Eppinger and Browning [26]. Originally developed as a network modelling tool, the DSM is a measure that captures the components of a system and their interactions. Various metrics can be deduced from this matrix [27–30], such as the number of components and connections within a system.

For decades these metrics have been used to evaluate key design features of digital systems, such as interoperability, modularity and complexity. For example, Presson [31] evaluates the interoperability by comparing the LOCs to the number of modules of a given software system. Land and Crnkovic [32] also use the LOCs metric to assess the maintainability of multiple digital systems integrated via different strategies. However, the integration costs in this case study are not quantified but merely estimated by the developers. Farid and McFarlane [33] adopt the DSM to evaluate the reconfigurability of manufacturing control systems. To assess the interoperability of object-oriented software, Saradhi and Sastry [34] analyse the cohesion and coupling of classes. The cohesion is computed by counting the pairs of connected methods within a class. The coupling is determined by counting the number of classes to which a given class is connected to.

2.3. Low-Cost Digital System Design Challenges

Low-cost digital systems differ in their design and implementation compared to conventional digital systems. Developers are faced with a number of challenges when designing such systems. Common design challenges identified in the literature revolve around extendability, standardised interfaces and open-source technologies. In particular, low-cost digital systems *should be* constructed as a sequence of *reusable* objects that combine both data and functions [35]. Moreover, selected information and communication *technologies need to be inexpensive* and *should easily integrate with external systems* by adopting standardised data formats and an open system architecture [36]. External systems include legacy devices and an existing IT infrastructure [37]. As part of several non-technical challenges, digital systems *should be easily accessible to non-expert users* [13], for example, by reducing the number of functions. Additionally, *hardware and software components should be open-source* while developers need to *achieve acceptable levels of reliability and safety* in an industrial setting [38]. For manufacturing, low-cost digital systems are likely to be created by *combining disparate components* through standardised interfaces, and such systems are likely to be *built successively* instead of as an integrated whole [16,39].

2.4. Approaches for Evaluating Designs Based on Architectures

There are numerous studies that qualitatively evaluate and compare architectures [14,40–43]. However, there are only a handful of approaches that assess the system designs that are directly derived from an architecture. The majority of evaluation studies rely on system architectures and use metrics to assess the performance of the resulting digital system. However, only a few studies quantitatively evaluate the architectural design itself. For example, Rahmani [44] evaluates architectures for electronic systems in aircraft

by using relative scores based on different system viewpoints. These scores capture the degree of contribution of different components towards a specific architectural feature. For manufacturing control, Chirn [45] evaluates the design effort and reconfigurability of a holonic manufacturing system compared to a centralised approach. The design effort is assessed by measuring the complexity in terms of transitions and arcs in Petric nets, and LOCs needed to build the control system. Apart from that, Brennan and Norrie [46] develop metrics to evaluate system architectures for a distributed autonomous control application in terms of its performance and structure. Specifically, the coupling between components is assessed by counting the number of messages sent and specifying the probabilities of decisions that components can make. Lindvall et al. [47] construct and assess the architecture of a client-server system in terms of maintainability by analysing the coupling between modules and classes within a module. Finally, Sant'Anna et al. [48] propose a framework to evaluate the modularity of software architecture designs based on concern-oriented metrics. In addition to the coupling and cohesion among architectural elements, they count the number of interfaces and operations of each component.

2.5. Research Gaps

There have been various studies which compare architectures for digital manufacturing. However, only a handful of approaches quantitatively assess the design of digital systems, which mainly focus on software systems derived from a system architecture. There are two main limitations of this research to date: (1) The majority of studies only perform a qualitative assessment of reference architectures, for example, by aligning their key architectural features. There are no approaches that evaluate the designs that are directly derived from a reference architecture. (2) Since there are no systematic design approaches to low-cost digital systems, there is a need for evaluating the effectiveness of designs for low-cost digital manufacturing systems that are based on one of the considered design approaches for digital manufacturing. The rationale behind this study is to evaluate the effectiveness of designs for low-cost digital manufacturing systems for the special case of designs that are directly based on a reference architecture.

3. Methodology

This section outlines the methodology of this study. A structured implementation comparison is performed to evaluate the effectiveness of designs for low-cost digital manufacturing systems. At first, we select a design approach for creating low-cost digital manufacturing designs, develop measures and metrics to evaluate the effectiveness of those designs, outline typical scenarios in the design of low-cost digital systems and implement these scenarios to create the evaluation testbed. We then quantify the effectiveness of designs by applying these measures and metrics to the implemented design scenarios of the evaluation testbed.

Concretely, the proposed methodology consists of five steps:

1. Select a design approach for digital manufacturing systems;
2. Develop measures and metrics to evaluate the effectiveness of designs relevant to the selected design approach;
3. Outline scenarios that typically occur when designing low-cost digital manufacturing systems;
4. Create an evaluation testbed by selecting specific low-cost digital solutions and using them to implement the design scenarios;
5. Evaluate the effectiveness of designs quantitatively by applying the measures and metrics to the implemented design scenarios.

Each of the steps of the proposed evaluation methodology is now described.

3.1. Selection of a Design Approach to Digital Manufacturing Systems

Several approaches to design digital manufacturing systems can be taken. Although any of the design approaches described in Section 2.1 can be used for the proposed evaluation

methodology, not every approach may be suitable for the design of particularly low-cost digital manufacturing systems. Therefore, the presented design approaches are analysed in terms of their capability to meet the different design challenges of low-cost digital systems described in Section 2.3 to identify those approaches that are most relevant for the design of low-cost digital systems.

3.2. Development of Evaluation Measures and Metrics

The effectiveness of designs for low-cost digital manufacturing systems is evaluated in terms of the effort required to design these systems. There are several factors that have an impact on the design effort when developing digital systems. However, not every impact factor depends on the selected design approach. Hence, it is necessary to only consider impact factors relevant for the selected design approach. Based on these impact factors, appropriate measures and metrics are developed to quantify the effectiveness of designs.

3.3. Selection of Low-Cost Digital Design Scenarios

For the design effectiveness evaluation, a number low-cost digital design scenarios are proposed. Due to the limited number of industrial deployments of systematically designed low-cost digital manufacturing systems, these scenarios are mainly based around the low-cost digital manufacturing system design challenges described in Section 2.3. Each scenario addresses a key design challenge of low-cost digital manufacturing systems, such as modularity and extendability. For example, a particularly low-cost sensor as part of a monitoring solution may fail more often than a more expensive sensor. An effective low-cost digital system design should thus not require much effort when replacing a system component. Apart from that, low-cost digital systems are likely to require evolving with the introduction of new production requirements [37]. Hence, developers should only be required to spend limited effort on design extensions, such as the addition of another feature or the integration of two digital systems.

Since these *design scenarios are biased towards particularly low-cost digital systems*, some design approaches are better suited to address these design challenges than others. For example, a highly reusable object-oriented design is likely to be more effective than an axiomatic design, since the latter has been developed for large complex systems which are beyond the scope of the low-cost digital systems considered in this study. Therefore, in addition to the proposed design challenges, we discuss how a low-cost design for typical digital manufacturing applications without a cost constraint can be realised. The first four scenarios include physical experiments where a baseline low-cost digital manufacturing system is built in a lab and modified depending on the scenario at hand, whereas the last scenario discusses other cases of low-cost digital manufacturing designs theoretically:

(1) Adding an extra feature to the system;
(2) Adding an additional (homogeneous) component to the system;
(3) Replacing a physical and virtual resource of the system;
(4) Integrating two systems;
(5) Other cases in manufacturing system designs:
    (a) A low-cost design of a demand-response application;
    (b) A low-cost design of a large-scale distributed digital system;
    (c) Exceptions in low-cost digital system designs.

3.4. Considered Low-Cost Digital Solutions for the Evaluation Testbed

Three low-cost digital solutions are selected to implement the aforementioned design scenarios, namely vibration monitoring, job tracking and digital job cards, because these are highly prioritised by manufacturing SMEs [13]. This section refers to digital systems under development as 'solutions' because they are assembled to solve a need for an SME. To ensure comparability, the main components of these digital solutions remain unchanged when applying the different design changes. The three digital solutions consist of the following components and functions:

- **Vibration monitoring** comprises a vibration sensor (Raspberry Pi Sense HAT) attached to a Raspberry Pi (Raspberry Pi 3B) and a web-based dashboard to show the vibration data (HTML, JS).
- **Job tracking** saves the location of current jobs, captures events of newly started jobs and visualises that information via a web-based dashboard. In addition to a barcode scanner (Tera 8100) which is connected to a Raspberry Pi (Raspberry Pi 3B) for tracking the jobs, it consists of a database for storing the job and location data (SQLite) and three web-based dashboards, one for each role on the shop floor (Django).
- **Digital job cards** supply operators at workstations with work order information, such as the quantity and type of product that needs to be produced. Operators scan barcodes of products or raw material that arrives at a workstation to receive the job card information from a database and display it on a web-based dashboard. Digital job cards rely on the same components as job tracking (Tera 8100, Raspberry Pi 3B, SQLite, Django).

3.5. Quantitative Evaluation of the Designs

To quantify the effectiveness of designs for low-cost digital manufacturing systems, a structured implementation comparison is performed. Specifically, the developed effectiveness measures and metrics are applied to the implemented low-cost digital design scenarios. In addition to the quantitative evaluation, other cases in the design of digital manufacturing systems that are typically not constrained by cost are discussed.

In the following two sections, the proposed evaluation methodology is carried out for the special case of designs that are directly based on a reference architecture. The first section captures the first four steps to develop the effectiveness measures and metrics and implement the design scenarios for creating the evaluation testbed. The second section performs the quantitative evaluation by applying the measures and metrics to the implemented design scenarios.

4. Development of the Evaluation Testbed

This section captures the first four steps of the proposed evaluation methodology. First, the presented design approaches are analysed in terms of their capability to meet the different low-cost digital design challenges to identify those relevant to low-cost digital systems. Although any of the presented design approaches can be used for the proposed methodology, this study performs the evaluation for the special case of designs that are directly based on reference architectures. Second, design effectiveness measures and metrics that are relevant to these reference architectures are developed. Third, the three selected low-cost digital solutions are used to implement the different design scenarios resulting in a testbed for the quantitative evaluation. Finally, the different designs are verified and validated.

4.1. A Case Study Using Reference-Architecture-Based Designs

Not all design approaches are capable of addressing the various design challenges of low-cost digital systems. The *tree-structured methodology* yields rigid system designs, which require much effort if design changes need to be made. Conversely, the *platform-* and *network-based methodologies* provide a high flexibility for connecting system components, thus enabling a successive development process. As most *architectures* are based on these methodologies, and they offer similar levels of flexibility. More detailed architectures additionally include standards, technology recommendations and implementation guidelines, such as the development of unified interfaces, which vastly reduces the required effort when designing low-cost digital systems containing disparate components. In contrast, the *axiomatic design approach* is not suitable for the design of low-cost systems, since its focus lies in large complex systems. However, this study is concerned with small digital systems with few functional requirements and components. Thus, adopting an axiomatic design approach would yield an undesirable design overhead. Finally, an *object-oriented approach*

is insufficient for designing low-cost digital manufacturing systems because it cannot be applied to hardware components.

This study uses the proposed evaluation methodology for the special case of designs that are directly based on reference architectures. *Reference architectures are models that provide a structured template solution with common terminology and support the design of digital systems through a systematic repeatable framework* [14]. For the structured implementation comparison, we select three candidate reference architectures relevant to low-cost digital systems, namely Shoestring [16], WoT [49] and PROSA/Erlang [50,51], since they have been compared previously [37]:

- **Shoestring** proposes two types of elements, namely *building blocks* and *service modules*. Building blocks provide base functionalities and well-defined interfaces by encapsulating an off-the-shelf technology, whereas service modules join several building blocks with wrappers to provide their functions and data at the network level.
- **WoT** abstracts any phyiscal or virtual resource as a web resource. These so called *things* consist of standardised metadata, which are processed by *consumers* to interact with those things.
- **PROSA** describes the roles and components of manufacturing systems via autonomous cooperating agents, so-called holons [52]. *Resource holons* contain a physical part representing a system component and an information processing part controlling this component, while *order holons* define a task in the manufacturing system and coordinate the operation.

Figure 1 illustrates how each reference architecture encapsulates system components into their respective architecture elements. For a monitoring system comprising a Raspberry Pi and a sensor, each *architectural element* (service module, thing, holon) provides an abstract view on these system *components* and provides their functionalities and data at a network level. The selected *reference architectures are likely to continue to evolve*. The evaluation conducted in this study captures the effectiveness of designs based on each reference architecture at a given moment of time. We acknowledge that the evaluation results may change with future developments. Additionally, using different reference architectures would have a significant impact on the evaluation results. For example, there are many highly abstract digital manufacturing reference architectures with only limited design and implementation guidance, such as RAMI 4.0 or IIRA [14]. These more abstract reference architectures would be much less effective in supporting the design of low-cost digital manufacturing systems compared to Shoestring, WoT or PROSA/Erlang.

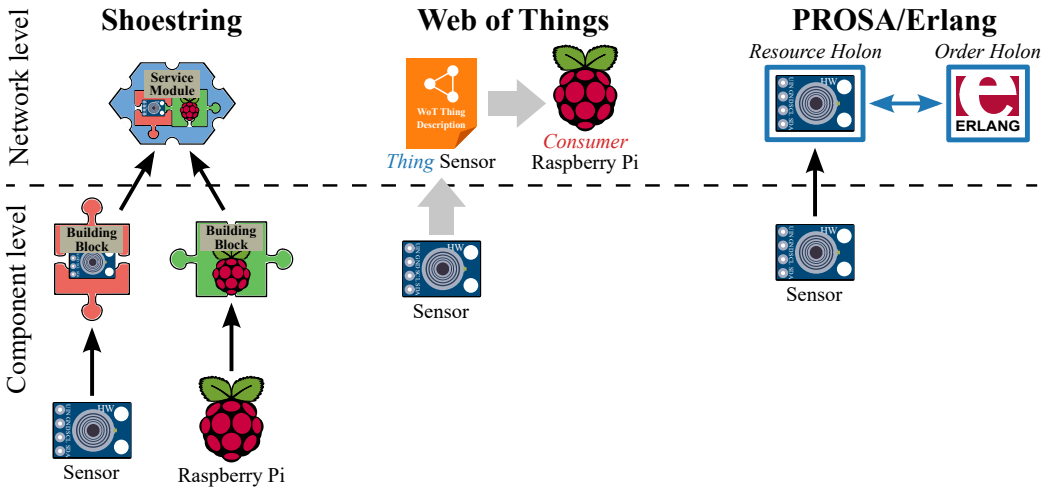

Figure 1. Example of how Shoestring, WoT and PROSA/Erlang encapsulate system components into their respective architecture elements (service modules, things, holons) [37].

4.2. Design Effectiveness Measures and Metrics

To assess the effectiveness of designs, the effort required to design a digital system is measured. Several factors have an impact on the design effort when adopting a reference architecture. These factors can be clustered into three areas, namely system, architecture, and designer. System dependent impact factors include performance requirements, such as latency, and data requirements, such as bandwidth and storage capacity. Architectural impact factors involve the design overhead caused by the architecture, the ease of integration of components and the technology a particular reference architecture is based on. Designers affect the design effort in terms of their skills. Specifically, compared to a novice, a skilled designer may perceive the effort of building a specific digital system as low. The time to develop a digital system may be affected by all three areas.

To assess the effectiveness of designs that are directly based on a reference architecture, we focus on architectural impact factors. These factors need to be *measurable* and *independent* from a specific system requirement or the designer. While measurability is necessary to quantify the required design effort, independence ensures an objective evaluation and comparison. In particular, design guidelines provided by a reference architecture may not always be strict and thus the required design effort can vary significantly across designs. In what follows, an overview of the architectural impact factors and selected measures and metrics is provided, which is summarised in Table 1. The measures and metrics have been validated through conducting an initial evaluation test. In this test, each selected measure and metric is applied to the three different baseline solution designs described in Section 3.4. If an evaluation measure or metric can be successfully applied and yields meaningful results, it is deemed to be validated.

Table 1. The developed design effectiveness measures and metrics for each architectural impact factor.

Impact Factor	Measure	Metrics
Architecture overhead	DSM	Size of architecture elements Pct. of architecture elements overlap No. architecture elements No. additional components No. connected components
Component integrability	DSM	No. system components No. auxiliary and interface components Size of interface components

Architecture overhead. The architecture overhead captures the effort of building the architectural elements and tailoring the digital system to connect the system components to those elements. It additionally provides insights on the complexity of the digital system imposed by the reference architecture. The effort of constructing the architecture elements is determined by counting the overall number of those elements for a given system, measuring their size by comparing the LOCs of these elements to the LOCs of the overall digital system and specifying their overlap, which describes how similar these elements are to one another. The tailoring effort to connect components to architecture elements is assessed by calculating the number of additional components, such as an extra HTTP server functioning as an intermediary between a component and an architecture element. The complexity imposed by the reference architecture is measured by counting the number of interactions (i.e., 1's) in the DSM.

A key benefit of using the DSM for these modular reference architecture designs is that it requires a clear separation between the different elements of a low-cost digital manufacturing system (e.g., hardware and software components, architecture elements, additional interface components) and the identification of all interactions between those elements. Other design evaluation measures, such as flowcharts, are not subject to such strict rules, which are likely to yield evaluation results that are not comparable across different reference architecture designs.

Component integrability. The component integrability measures the effort of integrating components with the architecture elements. Indicators for the component integrability are the overall number of components of the implemented system, which is determined by counting the number of rows/columns in the DSM, and the number and size of auxiliary and interface components. Such auxiliary components may be required if the elements of the reference architecture and the components of the digital system are implemented via different technologies, which is likely to increase the effort needed to build interfaces. For example, a given architecture element may rely on TCP/IP sockets, whereas a system component requires an HTTP interface. This interface disparity causes the designer to spend more time connecting the system components to the architecture elements. Apart from that, the size of interfaces is determined by computing the percentage of the interface LOCs compared to the overall size of the low-cost digital system.

4.3. Implementation of the Low-Cost Digital Design Scenarios

To create the testbed for the structured implementation comparison, the three selected low-cost digital solutions are used to implement the different design scenarios for each candidate reference architecture. For the implementation, a range of different low-cost (hardware and software) components are incorporated into the designs. Using a different set of low-cost components for the solution designs may only have a small effect on the evaluation results, since to be considered 'low-cost', selected components should satisfy the requirements derived from the various design challenges described in Section 2.3. For example, low-cost components should provide a commonly used interface, such that in case of a malfunction the effort of replacing them is low. An overview of the implemented design scenarios is shown in Table 2. Overall, there are 18 different designs. While the first three design scenarios are applied to vibration monitoring, the integration is performed for job tracking and digital job cards. The latter has been choosen to solve the needs of a timber doorset manufacturer in the UK as a result from an in-company requirements workshop:

Table 2. Implementation of the low-cost digital design scenarios. The first three scenarios are applied to vibration monitoring (VM), while the integration is performed for job tracking (JT) and digital job cards (DJC).

	Baseline	**Add Feature**	**Add Component**	**Replace Phys. Resource**	**Replace Virt. Resource**	**Integration**
Shoestring	VM	VM	VM	VM	VM	JT & DJC
WoT	VM	VM	VM	VM	VM	JT & DJC
PROSA/Erlang	VM	VM	VM	VM	VM	JT & DJC

(1) *Add a feature*: a threshold function is added for analysing the sensor data to detect the status of the machine. Shoestring captures this functionality with an additional architecture element, while PROSA/Erlang and WoT incorporate the function into the sensor component.
(2) *Add a component*: vibration monitoring is augmented by a noise filter and a graph visualisation. For each candidate reference architecture, these functions are added at the component level and do not yield new architecture elements.

(3) *Replace a resource*: the vibration sensor is replaced by a temperature sensor (physical) and a GUI is used instead of the dashboard (virtual). These replacements only require minor modifications of the interfaces and architecture elements for the three candidates.
(4) *Integration*: both job tracking and digital job cards share the same barcode scanner. The integration strategy relies on coordinators in combination with a common service bus and virtual data integration [37]: Shoestring designs these coordinators based on service wrappers. WoT only uses one client to manage the resources of both solutions. For PROSA/Erlang, order holons are added, which form a holarchy with the holons of the individual solutions.

4.4. Verification and Validation of Designs

Before the candidates can be evaluated, it is necessary to assess if the reference architectures have been adopted accurately and if the resulting digital systems work as expected, which is commonly known as *design verification and validation*. Each design is *verified* by assessing to which degree the designed system complies with the implementation guidelines and descriptions imposed by the reference architecture [53]. Architectural compliance is achieved when all features that have been designed based on a reference architecture are covered by and are in accordance with its specification. However, not all features need to be used to reach architectural compliance. This study assesses the architectural compliance (i.e., verifies the reference architecture designs) via a checklist of characteristics and implementation guidelines for each reference architecture.

Additionally, each design is *validated* by testing the functionality, interoperability and performance of the digital system as a result of adopting a particular design approach: the functionality is tested by checking the specifications of a digital system against its implemented features and functions. The interoperability is assessed by performing a series of tests to gauge certain interoperability aspects, such as the ease of information exchange between architectural elements and the consistency in terms of common programming styles and variable naming conventions. The performance of a digital system is tested by selecting certain parameters based on its specifications, such as the accuracy or range of a sensor, and testing the system against these parameters.

5. Evaluation of Designs

This section captures the last step of the proposed evaluation methodology. Specifically, the measures and metrics are applied to the implemented design scenarios to quantify the effectiveness of designs that are directly based on one of the candidate reference architectures for low-cost digital systems. We further discuss how a low-cost design can be achieved for other cases in digital manufacturing system designs.

5.1. Quantifying the Effectiveness of Reference Architecture Based Designs

The effectiveness of designs for low-cost digital manufacturing systems is evaluated in terms of the required design effort when adopting a reference architecture. To quantify the effectiveness of designs, we apply the measures and metrics to the implemented low-cost digital design scenarios for each candidate reference architecture. For illustration, Tables 3 and 4 show the DSMs of WoT and PROSA/Erlang for the scenario where the vibration sensor is replaced. The evaluation is summarised in Table 5. Additionally, Figure 2 visualises the evaluation data in form of a heat map. The colours in the heat map indicate the relative design effort when adopting one of the three candidate reference architectures for a particular design scenario. As indicated in Section 4.1, the evaluation captures the effectiveness of designs based on each reference architecture at a given moment in time. The results may change with future developments.

Table 3. DSM of the WoT vibration monitoring design when replacing a physical resource.

Design Structure Matrix WoT Replace Phys. Resource	**Thing Description Server**	**Client**	**Temperature Sensor**	**Dashboard Server**	**Dashboard**
Thing Description Server	x	1	1		
Client	1	x			
Temperature Sensor			x		
Dashboard Server	1			x	
Dashboard				1	x

Table 4. DSM of the PROSA/Erlang vibration monitoring design when replacing a physical resource.

Design Structure Matrix PROSA/Erlang Replace Phys. Resource	**Start Node**	**Order Holon**	**Sensor Resource Holon**	**Temperature Sensor**	**Dashboard Resource Holon**	**Dashboard Server**	**Dashboard**
Start Node	x						
Order Holon	1	x	1		1		
Sensor Resource Holon	1	1	x	1			
Temperature Sensor				x			
Dashboard Resource Holon	1	1			x		
Dashboard Server					1	x	
Dashboard						1	x

Table 5. Effectiveness evaluation of the designs based on Shoestring, WoT and PROSA/Erlang.

	Baseline	Add Feature	Add Component	Replace Phys. Resource	Replace Virt. Resource	Integration
			Shoestring			
Size of architecture elements	53.2%	57.4%	44.1%	53.1%	76.4%	18.4%
Pct. of architecture elements overlap	98.6%	98.6%	98.6%	98.6%	98.6%	92.3%
No. architecture elements	2	3	2	2	2	9
No. additional components	0	0	0	0	0	1
No. connected components	4	8	6	4	3	18
No. system components	5	7	6	5	4	16
No. auxiliary and interface components	0	0	0	0	0	0
Size of auxiliary and interface components	4.2%	5.0%	5.6%	4.2%	6.0%	1.5%
			WoT			
Size of architecture elements	37.1%	40.9%	35.2%	36.1%	67.8%	9.2%
Pct. of architecture elements overlap	9.3%	14.5%	13.6%	15.5%	15.3%	21.7%
No. architecture elements	3	3	3	3	3	9
No. additional components	0	0	0	0	0	0
No. connected components	5	5	7	5	4	24
No. system components	5	5	6	5	4	16
No. auxiliary and interface components	0	0	0	0	0	5
Size of auxiliary and interface components	1.3%	1.8%	2.4%	1.3%	1.8%	3.8%
			PROSA/Erlang			
Size of architecture elements	56.7%	55.3%	50.2%	56.5%	74.9%	17.6%
Pct. of architecture elements overlap	41.7%	41.7%	41.7%	41.7%	41.7%	44.3%
No. architecture elements	3	3	3	3	3	11
No. additional components	0	0	0	0	1	0
No. connected components	10	10	10	10	10	54
No. system components	7	7	7	7	7	23
No. auxiliary and interface components	0	0	0	0	0	4
Size of auxiliary and interface components	9.1%	8.9%	8.1%	9.1%	12.6%	5.6%

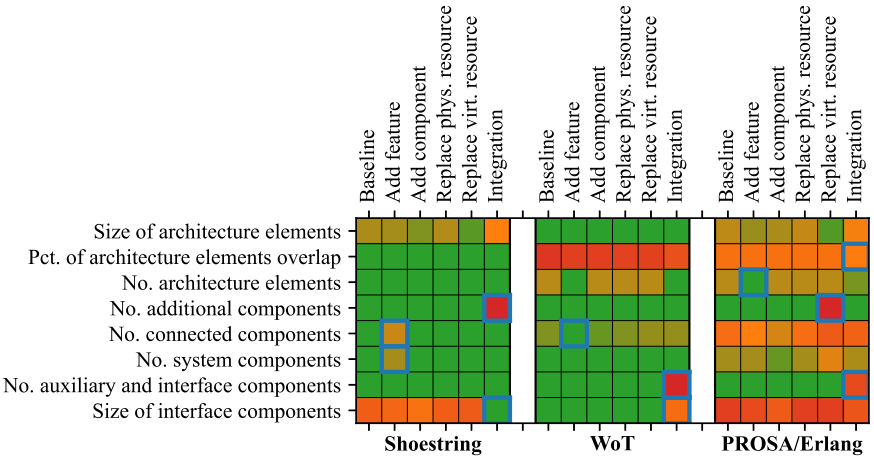

Figure 2. Heat map of the effectiveness evaluation of designs based on Shoestring, WoT and PROSA/Erlang. The colours indicate the relative design effort for the three candidates (green: low; orange: medium; red: high). The outliers (blue) represent cases where the design effort is notably different compared to the other design scenarios.

5.1.1. Architecture Overhead

WoT includes a software framework supporting the creation of its architecture elements, which vastly reduces the required LOC. However, the resulting source code is highly heterogeneous, as sections of code that are equal across multiple systems have been embedded in this framework. Thus, developers only need to write the code which is custom for each architecture element, such as IDs and descriptions, which yields a small architecture element overlap. Compared to Shoestring, for which the architecture elements have a high degree of similarity, the structural differences between resource and order holons yield a smaller overlap. Additionally, Shoestring has no coordinator that is central to a single system. Consequently, WoT and PROSA/Erlang have more architecture elements for each design than Shoestring. Moreover, the candidates rarely require additional components for the conducted design scenarios. For the integration scenario, Shoestring includes a schema to map data across both systems, while for PROSA/Erlang and WoT coordinators handle this logical mapping. When replacing the web-based dashboard with a GUI component, PROSA/Erlang requires an additional server which functions as a data repository. While this additional component can be avoided by modifying the interface of the architecture element, it would yield more changes in the source code than simply adding such an intermediary repository. Apart from that, WoT and PROSA/Erlang have more interactions among components than Shoestring, since their coordinators serve as a gateway, which inherently leads to more messages sent. Conversely, the service modules of Shoestring directly pass information to other service modules upon request. An exception forms the added analysis feature, because Shoestring requires such features to be wrapped into an additional architecture element. For the other candidates, this feature is embedded into the architecture element of the sensor.

5.1.2. Component Integrability

There are no differences between WoT and Shoestring when comparing the overall number of system components. Since WoT has less strict implementation guides, all resources are exposed by a single server component, which reduces the overall design effort required for its designs. Conversely, PROSA/Erlang requires all holons to be separate system components. As indicated above, the analysis feature for Shoestring creates an additional architecture element and system component. Auxiliary and interface components are only necessary for the integration scenario for WoT and PROSA/Erlang, because the individual solutions require additional WebSockets and servers to connect the disparate components to the architecture elements. Shoestring does not require such auxiliary and interface components for the integration scenario because it is based on the same pro-

gramming language as the two integrated solutions, which vastly simplifies connecting architecture elements to individual components. Whereas PROSA/Erlang requires designers to use Erlang/OTP for the development of its architecture elements, a WoT design could also be created using the same programming language as Shoestring and the two digital solutions, which would reduce the number of required auxiliary and interface components. However, this feature of WoT is outdated, and as a result, developers risk a longer design process. Similar to the size of architecture elements, WoT outperforms Shoestring and PROSA/Erlang because its software framework allows for a minimal interface. Specifically, two LOCs are needed for sending an HTTP request and processing its response. By comparison, the PROSA/Erlang relies on TCP/IP sockets for its designs, while Shoestring uses an additional communication layer to send messages from the component to the network level. Therefore, the size of the interface components is much larger for these two candidates. An exception forms the integration scenario because the additional interface components required for the WoT and PROSA/Erlang designs carries over to the overall size of the interface components.

The overall design effectiveness can now be determined by comparing the columns (i.e., the metric values for each design scenario) in Figure 2 and Table 5. In summary, the designs based on Shoestring are more effective compared to those based on PROSA/Erlang and WoT, since they require the least design effort for the considered low-cost digital design scenarios. However, compared to WoT, Shoestring suffers from a more complex design in some scenarios due to its focus on reusability. Additionally, due to its software framework, WoT requires the least effort when designing the interfaces for connecting the system components to the architecture elements.

5.2. Other Cases in Manufacturing System Designs

The effectiveness of designs that are directly based on a reference architecture has been evaluated through a structured implementation comparison based on typical low-cost digital design scenarios. However, WoT and PROSA/Erlang have not strictly been designed for these scenarios. Therefore, we additionally discuss how a low-cost design can be achieved for cases in digital manufacturing system designs that are not typically constrained by cost. Specifically, we investigate how a low-cost design based on Shoestring can be developed for a typical WoT and PROSA/Erlang application, and discuss how the required design effort may change for exceptions in low-cost digital system designs.

5.2.1. A Low-Cost Design of a Demand–Response Application

PROSA was originally developed for demand–response applications, such as a customer order management solution [50]. Specifically, PROSA decouples logistical aspects from technical ones and allows for more advanced hybrid control algorithms. These capabilities are not required for the low-cost digital systems considered in this study, and thus, they have not been captured by the selected design scenarios. Hence, we investigate how Shoestring could be adopted to design such a demand-response application.

As an example we consider a packaging cell for gift boxes, which consists of a robot arm for picking up boxes from a buffer and packaging them, and a gantry robot to move packed boxes to a shelf [54]. A Shoestring design could be as follows: both robots, including the control functions, are controlled via actuation service modules. The control commands are triggered by sensors wrapped as data collection service modules. In contrast, PROSA would encapsulate the robots into resource holons and represent the current state of a given box by a product holon. Order holons would characterise a specific customer order and exchange relevant process information between the resource and product holons. Compared to PROSA, Shoestring handles customer orders as events and is unaware of the product state at any given time, which impedes the scheduling and optimisation of the process. Additionally, more architectural elements are required for Shoestring, yielding a more complex system. We expect that PROSA entities are better equipped for such scenarios [55]. Specifically, Shoestring would need an additional scheduling service module,

while holons can negotiate their schedule by themselves. Holons are more adaptable since they find a schedule locally, while the scheduling service module of the Shoestring design would need to solve this problem globally.

5.2.2. A Low-Cost Design of a Large-Scale Distributed System

WoT was developed to enable interoperability across IoT application domains [49] and is particularly well-suited for large-scale distributed systems. Specifically, a 'web' of things is constructed by semantically linking information models, which enables the development of an ontology and discovery mechanisms. Such capabilities are not required for the small-scale low-cost digital systems considered in this study, and thus, they have not been represented by the selected design scenarios. Hence, we discuss how Shoestring could yield such capabilities for the design of a large-scale distributed system.

For illustration, we consider a smart factory with various stationary and mobile machines connected to a number of management applications. While the Shoestring design of small monitoring and tracking applications is likely to be similar to those based on WoT, more complex machines, such as a robots, would need to be represented by a complete Shoestring system. As described earlier, a robot could be represented by an actuation and data collection service module. Conversely, WoT only requires a single architecture element for these resources. Therefore, a Shoestring design would be less effective than a design based on WoT for this scenario. Furthermore, security becomes an issue of concern for large-scale systems, as systems across domains may not share the same network. Contrary to WoT, Shoestring does not offer security features.

5.2.3. Exceptions in Low-Cost Digital System Designs

There are a handful of exceptional cases when designing low-cost digital systems, such as the addition of heterogeneous components, multiple components residing in the same location and the development of a large number of homogeneous systems. In what follows, we discuss how these cases impact the effectiveness of designs for each candidate reference architecture:

- *Addition of heterogeneous components.* Low-cost systems are likely to consists of disparate components [16,39]. Adding such heterogeneous components may yield different interface sizes between the architecture elements and components. For example, a sensor type other than the Raspberry Pi Sense HAT may require a more complex interface. While some sensors include an API, other types may require a listener to the port these types are connected to. However, these changes only have a small effect on the overall design effort required, as they need to be created for each system irregardless of the selected reference architecture.
- *Multiple components in the same location.* Applications for which multiple components reside in the same physical location are typically monitoring solutions. Whereas larger distributed monitoring applications are likely to consist of multiple sensing locations, this case considers all sensors to reside in only one location. There are two main effects on the required design effort: First, less hardware components are needed since all sensors can be attached to a single computational device. Second, the data aggregation can be implemented at a component level, thus reducing the network traffic. There are different impacts on design based on the three candidates: For Shoestring, instead of having a data collection service module for each location, all sensors can be represented by a single service module. Conversely, WoT and PROSA/Erlang imply wrapping each sensor into a separate architecture element. Therefore, the designs based on Shoestring are likely to be more effective than WoT and PROSA/Erlang since they require fewer architectural elements, system components and number connected elements. However, PROSA/Erlang enables the aggregation of multiple sensor holons, which results in a more streamlined design. As mentioned above, the size of the interfaces remains equal across the candidates since the interfaces have to be built for each sensor independent from the chosen reference architecture.

- *A large number of homogeneous systems.* Given the limited resources of SMEs, low-cost digital systems may be developed successively [13,16,20]. Companies may start with an initial tracking or monitoring system for one workstation and proceed with additional systems once it has been successfully deployed. This case covers the successive design of a large number of homogeneous systems. For illustration, we consider the design of 30 monitoring solutions. Due to their homogeneity, the effort required to design subsequent solutions is vastly reduced since the design of the initial solution can simply be reused. In particular, the modularity of the candidate reference architecture is a key driver for reusability. However, as the candidates have similar levels of modularity, the effectiveness of the designs does not differ significantly between the reference architectures. Additionally, with an increasing number of systems, developers need to take the system performance into consideration, such as latency and throughput.

In summary, although a low-cost design based on Shoestring is feasible for a typical WoT and PROSA/Erlang application, its design would be less effective. Specifically, the designs based on Shoestring would be larger and more complex to obtain the required functionalities for each case. However, due to its focus on reusability, a design based on Shoestring is more effective compared to WoT and PROSA/Erlang in cases where multiple similar low-cost digital systems need to be designed.

6. Discussion

This section discusses the development of the evaluation testbed and quantitative evaluation of the designs based on the three selected candidate reference architectures. First, we compare the proposed metrics-based evaluation methodology with existing approaches to evaluate architectural designs and describe the limitations of our study. We then discuss the importance of the proposed metrics for practitioners and examine how these metrics can be used to predict changes in future designs. Finally, we discuss how the metrics need to be modified when a different design approach is selected for evaluating the effectiveness of low-cost digital system designs.

6.1. Comparison with Existing Approaches for Evaluating Architectural Designs

This study has performed a structured implementation comparison to quantify the effectiveness of designs for low-cost digital systems that are directly based on reference architectures. The effectiveness is evaluated in terms of the effort required to design a low-cost digital system. Compared to the studies that focus on a feature comparison and mapping and alignment of reference architectures [14,40–43], this study concentrates on the designs as a result of adopting a reference architecture and analyses their effectiveness based on a set of measures and metrics. In contrast to Rahmani [44], we focus on reference architectures and develop metrics which are relevant to the chosen design approach but independent from a particular system design. Similar to Chirn [45], this study relies on the LOCs metric to assess the design effort of architectural designs. Compared to Brennan and Norrie [46], the evaluation conducted in this study is centred on the design and does not capture the performance of resulting digital systems. Contrary to Lindvall et al. [47], this study does not analyse the coupling or cohesion since these metrics can only be applied to software systems and the types of systems considered in this study also consist of hardware components. Finally, similar to Sant'Anna et al. [48], we count the number of interfaces and interactions among components in this study for assessing the system design. It is noteworthy to mention that the evaluation approach does not necessarily need to involve metrics being applied to the final design. For example, an alternative approach may involve applying the metrics before and during the design process of a digital system to facilitate making decisions early in the design process [56].

6.2. Limitations of the Proposed Evaluation Methodology

This study is subject to three limitations: First, we only focus on the design of particularly low-cost digital systems. Different design scenarios may need to be considered when

evaluating the designs that are not constrained by cost. Second, these scenarios are biased towards the Shoestring reference architecture because we have evaluated designs from different reference architectures in the domain Shoestring was developed for, and there are only few detailed operational reference architectures that can potentially realise digital systems at low cost. Third, the proposed methodology is only applied to evaluate designs based on the same design approach.

6.3. Importance of Evaluation Metrics for Practitioners

The proposed evaluation metrics may not be equally important to practitioners since they may be focused on different aspects when designing a digital system depending on its specifications and requirements. For example, to ensure a reliable system performance, reducing the complexity in terms of overall interactions between components may be more important to a designer than decreasing the size of architecture elements, because the risk of faults increases with the number of interactions of a digital system. Independent from the designer, the maturity of a reference architecture may affect the importance of specific metrics for practitioners as well. With increasing maturity, reference architectures may embed the source code of commonly used architecture elements into reusable software modules, which might evolve to a software framework, such as the reference implementation of WoT. Consequently, the size of these architecture elements may only have a small effect on the required design effort when adopting a more mature reference architecture.

6.4. Predictability of Evaluation Metrics to Future Designs

The proposed metrics cannot only be used to assess the effectiveness of existing designs, but they may also influence changes in future designs and in the design approach overall. In particular, outliers in the design effort evaluation data as shown in Figure 2 can provide valuable insights for developers who aim to improve an existing reference architecture or construct a new one. *Outliers are cases in datasets that behave differently from the majority of data* [57]. There are several ways to identify outliers. This study detects outliers by creating an upper and lower boundary of $\pm 2\sigma$ for each metric (i.e., for each row in the heat map).

An outlier in the evaluation data represents a significant difference in the required design effort for a specific low-cost digital design scenario when comparing the three candidate reference architectures. It reveals strengths and shortcomings in the design process imposed by a reference architecture and can therefore be used to improve the overall design approach. Specifically, to increase the effectiveness of reference architecture based designs, the architectural features that are connected to such outliers need to be modified or their negative effect on the design effort needs to be compensated by adding or modifying a different feature. Developers can predict the effectiveness (i.e. required design effort) of future designs by analysing these evaluation outliers:

- Compared to WoT and PROSA/Erlang, Shoestring adds further complexity to a system design by wrapping features, such as data analysis, into a separate architecture element. However, this overhead could facilitate the design process of new digital systems since Shoestring allows for more architecture elements to be reused. Developers have to consider this trade-off when designing digital systems.
- The share-nothing policy of Erlang/OTP leads to fault-tolerant systems but may also increase their complexity. For example, when replacing the web-based dashboard with a GUI, an additional component is required that serves as an intermediate data repository for PROSA/Erlang. This additional component cannot be avoided without major modifications of the system design. Therefore, developers have to consider that the required design effort varies for different types of resources and components, and it may be useful to spend more time on selecting components to reduce the overall number of design steps.

6.5. Development of Metrics for Other Design Approaches

As mentioned in Section 3, the proposed evaluation methodology can also be applied to other digital manufacturing design approaches, which would require the development

of new measures and metrics that are relevant to these approaches. For example, in contrast to a reference architecture based design, an object-oriented system design only includes software components. Since for software systems the size of the interfaces are likely to be smaller compared to digital systems that also comprise hardware components, it only has a small effect on the required design effort. Thus, such metrics may be less relevant for evaluating the effectiveness of object-oriented designs.

If the selected design approach is similar to a reference architecture based approach, the measures and metrics developed in this study can simply be reused. For instance, when comparing an evaluation of designs based on a system architecture and reference architecture, developers are faced with similar design challenges, such as developing the interfaces to system components and connecting them to the elements of the system architecture.

It is important to note that *the proposed methodology is used to evaluate the effectiveness of designs based on a single design approach*. To compare the effectiveness of designs based on two different design approaches, developers need to focus on the key differences between those two approaches. For example, to compare the effectiveness of designs based on a reference architecture and system architecture, the resulting metrics may capture the reusability of components when developing subsequent systems since reference architectures provide generally more support for a successive development of digital system than system architectures [37].

7. Conclusions

This study has evaluated the effectiveness of designs for low-cost digital manufacturing systems in terms of the effort required to design such systems. The proposed methodology is used for the special case of designs that are directly based on reference architectures. The evaluation is based on a structured implementation comparison between three candidate reference architectures relevant to low-cost digital systems. The main contribution of this study is the development of a systematic means for quantifying the effectiveness of designs based on a particular design approach.

The evaluation metrics and design scenarios offer sufficient granularity to assess the effectiveness of designs and provide insights into the shortcomings of a reference architecture based low-cost digital system design. Each candidate reference architecture suffers from different drawbacks when analysing the effort required to design a digital system. In particular, practitioners need to consider the various trade-offs when selecting a reference architecture, such as a more complex system design as opposed to an increased reusability of architectural elements. However, since the candidates are likely to continue to evolve, the evaluation results may change with future developments.

Four key issues are subject to future study: first, there is a need to evaluate the effectiveness of designs based on a different design approach, such as an object-oriented design approach. Apart from that, further insights can be gained by comparing the evaluation results of this study with those based on other design approaches. Second, a key driver for reducing the effort of designing low-cost digital systems is the reusability of the selected design approach. However, conclusions regarding the reusability of a design approach can only be drawn when evaluating a large number of digital system deployments, which is beyond the scope of this study. Third, while this study has discussed how the evaluation metrics indicate changes in future designs, more work is required to develop a proper prediction model. Finally, instead of concentrating on designs based on different reference architectures, the methodology could be applied to designs based on different versions of the same reference architecture, thus enabling version control. In this way, developers can verify if the changes made to a reference architecture have improved the design process.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app132312618/s1.

Author Contributions: Conceptualization, J.K., G.H., A.M. and D.M.; Methodology, J.K., G.H., A.M. and D.M.; Software, J.K.; Data curation, J.K.; Writing–original draft, J.K.; Writing–review & editing, G.H., A.M. and D.M.; Supervision, D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Engineering and Physical Sciences Research Council (EPSRC: EP/R032777/1).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in the Supplementary Materials.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gregory, S.A. *The Design Method*; Springer US: Berlin/Heidelberg, Germany, 1966.
2. Farid, A.M.; Suh, N.P. *Axiomatic Design in Large Systems*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016.
3. Moses, J. Architecting Engineering Systems. In Proceedings of the First Workshop on Philosophy and Engineering, Delft, The Netherlands, October 2007.
4. Clements, P.C. A Survey of Architecture Description Languages. In Proceedings of the 8th International Workshop on Software Specification and Design, Schloss Velen, Germany, 22–23 March 1996.
5. Rentsch, T. Object Oriented Programming. *ACM SIGPLAN Not.* **1982**, *17*, 51–57. [CrossRef]
6. Gamma, R.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns—Elements of Reusable Object-Oriented Software*; Prentice Hall: Hoboken, NJ, USA, 1995.
7. Narayanan, A.; Kanyuck, A.; Gupta, S.K.; Rachuri, S. Machine condition detection for milling operations using low cost ambient sensors. In Proceedings of the ASME International Manufacturing Science and Engineering Conference (MSEC2016), Blacksburg, VA, USA, 27 June–1 July 2016.
8. Aghenta, L.O.; Iqbal, M.T. Low-cost, open source IoT-based SCADA system design using thinger.IO and ESP32 thing. *Electronics* **2019**, *8*, 822. [CrossRef]
9. Cheddadi, Y.; Cheddadi, H.; Cheddadi, F.; Errahimi, F.; Es-sbai, N. Design and implementation of an intelligent low-cost IoT solution for energy monitoring of photovoltaic stations. *SN Appl. Sci.* **2020**, *2*, 1165. [CrossRef]
10. Sabadoti, V.D.; Miano, A.C.; Augusto, P.E.D. Automation of a Mattson Bean Cooker: A simple and a low-cost approach. *J. Food Process. Preserv.* **2020**, *44*, e14769. [CrossRef]
11. Kaiser, J.; Terrazas, G.; McFarlane, D.; de Silva, L. Towards low-cost machine learning solutions for manufacturing SMEs. *AI Soc.* **2021**, 1–7. [CrossRef]
12. Vuković, M.; Jorg, O.; Hosseinifard, M.; Fantoni, G. Low-Cost Digitalization Solution through Scalable IIoT Prototypes. *Appl. Sci.* **2022**, *12*, 8571. [CrossRef]
13. Schönfuß, B.; McFarlane, D.; Hawkridge, G.; Salter, L.; Athanassopoulou, N.; de Silva, L. A catalogue of digital solution areas for prioritising the needs of manufacturing SMEs. *Comput. Ind.* **2021**, *133*, 103532. [CrossRef]
14. Kaiser, J.; McFarlane, D.; Hawkridge, G.; André, P.; Leitão, P. A review of reference architectures for digital manufacturing: Classification, applicability and open issues. *Comput. Ind.* **2023**, *149*, 103923. [CrossRef]
15. Rojko, A. Industry 4.0 concept: Background and overview. *Int. J. Interact. Mob. Technol.* **2017**, *11*, 77–90. [CrossRef]
16. McFarlane, D.; Ratchev, S.; Thorne, A.; Parlikad, A.K.; de Silva, L.; Schönfuß, B.; Hawkridge, G.; Terrazas, G.; Tlegenov, Y. Digital Manufacturing on a Shoestring: Low Cost Digital Solutions for SMEs. In Proceedings of the Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future, Valencia, Spain, 3–4 October 2019.
17. Gunasekaran, A. Implementation of computer-integrated manufacturing: A survey of integration and adaptability issues. *Int. J. Comput. Integr. Manuf.* **1997**, *10*, 266–280. [CrossRef]
18. Park, K.T.; Nam, Y.W.; Lee, H.S.; Im, S.J.; Noh, S.D.; Son, J.Y.; Kim, H. Design and implementation of a digital twin application for a connected micro smart factory. *Int. J. Comput. Integr. Manuf.* **2019**, *32*, 596–614. [CrossRef]
19. Ryalat, M.; ElMoaqet, H.; AlFaouri, M. Design of a Smart Factory Based on Cyber-Physical Systems and Internet of Things towards Industry 4.0. *Appl. Sci.* **2023**, *13*, 2156. [CrossRef]
20. Hawkridge, G.; McFarlane, D.; Kaiser, J.; de Silva, L.; Terrazas, G. Designing Shoestring Solutions: An Approach for Designing Low-Cost Digital Solutions for Manufacturing. In Proceedings of the Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future, Cluny, France, 18–19 November 2021.
21. Damianos, L.; Hirschman, L.; Kozierok, R.; Kurtz, J.; Greenberg, A.; Walls, K.; Laskowski, J.S. Evaluation for collaborative systems. *ACM Comput. Surv.* **1999**, *31*, 2-es–15-es. [CrossRef]
22. Fenton, N.; Martin, N. Software Metrics: Roadmap. In Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 4–11 June 2000.

23. Poulin, J.S. Measuring software reusability. In Proceedings of the 3rd International Conference on Software Reuse, Rio de Janeiro, Brazil, 1–4 November 1994.
24. Mccabe, T.J. A Complexity Measure. *IEEE Trans. Softw. Eng.* **1976**, *4*, 308–320. [CrossRef]
25. Stevens, W.P.; Myers, G.J.; Constantine, L.L. Structured design. *IBM Syst. J.* **1974**, *13*, 115–139. [CrossRef]
26. Eppinger, S.D.; Browning, T.R. *Introduction to Design Structure Matrix Methods*; MIT Press: Cambridge, UK, 2012.
27. Sosa, M.E.; Eppinger, S.D.; Rowles, C.M. Identifying modular and integrative systems and their impact on design team interactions. *J. Mech. Des. Trans.* **2003**, *125*, 240–252. [CrossRef]
28. Martin, M.V.; Ishii, K. Design for variety: Developing standardized and modularized product platform architectures. *Res. Eng. Des.* **2002**, *13*, 213–235. [CrossRef]
29. Gershenson, J.K.; Prasad, G.J.; Zhang, Y. Product modularity: Measures and design methods. *J. Eng. Des.* **2004**, *15*, 33–51. [CrossRef]
30. Huang, C.C.; Kusiak, A. Modularity in design of products and systems. *IEEE Trans. Syst. Man Cybern. Syst.* **1998**, *28*, 66–77. [CrossRef]
31. Presson, P. Software metrics and interoperability. In Proceedings of the 4th Computers in Aerospace Conference, Hartford, CT, USA, 24–26 October 1983.
32. Land, R.; Crnkovic, I. Software Systems Integration and Architectural Analysis—A Case Study. In Proceedings of the IEEE International Conference on Software Maintenance (ICSM), Amsterdam, The Netherlands, 22–26 September 2003.
33. Farid, A.M.; McFarlane, D.C. A Design Structure Matrix Based Method for Reconfigurability Measurement of Distributed Manufacturing Systems. *Int. J. Intell. Control. Syst.* **2007**, *1*, 118–129.
34. Saradhi, M.V.V.; Sastry, B.R. A Quality Indicator for Software Interoperability. *Int. J. Eng. Sci.* **2010**, *2*, 2587–2594.
35. Roşca, D.; Bănică, L.; Sîrbu, M. Building Successful Information Systems-a Key for Successful Organization. *Econ. Appl. Inform.* **2010**, *1*, 101–108.
36. Filip, F.G. Designing and Building Modern Information Systems; A Series of Decisions to Be Made. *Comput. Sci. J. Mold.* **2011**, 4–13.
37. Kaiser, J.; Hawkridge, G.; McFarlane, D.; Schnicke, F.; Kruger, K. Effective integration of low-cost digital manufacturing systems: A reference architecture driven approach. *Int. J. Comput. Integr. Manuf.* 2023, *submitted*.
38. Hoxha, V.; Bula, I.; Muzafer, S.; Hajrizi, E. Cost-Oriented Open Source Automation Potential Application in Industrial Control Applications. *IFAC Proc. Vol.* **2016**, *49*, 212–214. [CrossRef]
39. Erbe, H.H. Low cost intelligent automation in manufacturing. *IFAC Proc. Vol.* **2002**, *35*, 373–378. [CrossRef]
40. Pedone, G.; Mezgár, I. Model similarity evidence and interoperability affinity in cloud-ready Industry 4.0 technologies. *Comput. Ind.* **2018**, *100*, 278–286. [CrossRef]
41. Fraile, F.; Sanchis, R.; Poler, R.; Ortiz, A. Reference Models for Digital Manufacturing Platforms. *Appl. Sci.* **2019**, *9*, 4433. [CrossRef]
42. Nakagawa, E.Y.; Antonino, P.O.; Schnicke, F.; Capilla, R.; Kuhn, T.; Liggesmeyer, P. Industry 4.0 reference architectures: State of the art and future trends. *Comput. Ind. Eng.* **2021**, *156*, 107241. [CrossRef]
43. Pivoto, D.G.S.; de Almeida, L.F.F.; da Rosa Righi, R.; Rodrigues, J.J.P.C.; Lugli, A.B.; Alberti, A.M. Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review. *J. Manuf. Syst.* **2021**, *58*, 176–192. [CrossRef]
44. Rahmani, S. An objective methodology for definition and evaluation of advanced avionics architectures. In Proceedings of the 7th Computers in Aerospace Conference, Monterey, CA, USA, 3–5 October 1989.
45. Chirn, J. Developing a Reconfigurable Manufacturing Control System—A Holonic Component-Based Approach. Ph.D. Thesis, University of Cambridge, Cambridge, UK, 2001.
46. Brennan, R.W.; Norrie, D.H. Metrics for evaluating distributed manufacturing control systems. *Comput. Ind.* **2003**, *51*, 225–235. [CrossRef]
47. Lindvall, M.; Tvedt, R.T.; Costa, P. An Empirically-Based Process for Software Architecture Evaluation. *Empir. Softw. Eng.* **2003**, *8*, 83–108. [CrossRef]
48. Sant'Anna, C.; Figueiredo, E.; Garcia, A.; Lucena, C.J.P. On the Modularity of Software Architectures: A Concern-Driven Measurement Framework. In Proceedings of the Software Architecture, First European Conference (ECSA), Madrid, Spain, 24–26 September 2007.
49. Lagally, M.; Matsukura, R.; McCool, M.; Toumura, K. Web of Things (WoT) Architecture 1.1—W3C Editor's Draft. 27 May 2021. Available online: https://w3c.github.io/wot-architecture/ (accessed on 27 May 2021).
50. Van Brussel, H.; Wyns, J.; Valckenaers, P.; Bongaerts, L.; Peeters, P. Reference architecture for holonic manufacturing systems: PROSA. *Comput. Ind.* **1998**, *37*, 255–274. [CrossRef]
51. Kruger, K.; Basson, A. Erlang-based control implementation for a holonic manufacturing cell. *Int. J. Comput. Integr. Manuf.* **2017**, *30*, 641–652. [CrossRef]
52. Koestler, A. *The Ghost in the Machine*; Hutchinson: London, UK, 1967.
53. Michael, J.B.; Riehle, R.; Shing, M. The verification and validation of software architecture for systems of systems. In Proceedings of the IEEE International Conference on System of Systems Engineering (SoSE), Albuquerque, NM, USA, 30 May–3 June 2009.
54. Covanich, W.; McFarlane, D.; Brusey, J.; Farid, A.M. Integrating a New Machine into an Existing Manufacturing System. In Proceedings of the 5th IEEE International Conference on Industrial Informatics, Vienna, Austria, 23–27 June 2007.
55. De Swert, K.; Valckenaers, P.; German, B.S.; Verstraete, P.; Van Brussel, H. Coordination and control for railroad networks inspired by manufacturing control. In Proceedings of the IEEE Workshop on Distributed Intelligent Systems - Collective Intelligence and Its Applications, Prague, Czech Republic, 15–16 June 2006.

56. Penadés-Plà, V.; García-Segura, T.; Martí, J.V.; Yepes, V. A Review of Multi-Criteria Decision-Making Methods Applied to the Sustainable Bridge Design. *Sustainability* **2016**, *8*, 1295. [CrossRef]
57. Rousseeuw, P.J.; Hubert, M. Anomaly detection by robust statistics. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1236. [CrossRef]