RESEARCH ARTICLE

Effective integration of low-cost digital manufacturing systems: a reference architecture driven approach

J. Kaiser[a], G. Hawkridge[a], D. McFarlane[a], F. Schnicke[b] and K. Kruger[c]

[a] University of Cambridge, 17 Charles Babbage Rd, Cambridge CB3 0FS, United Kingdom
[b] Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
[c] Stellenbosch University, Cnr Banhoek Road & Joubert Street, Stellenbosch 7600, South Africa

ARTICLE HISTORY
Compiled January 5, 2024

ABSTRACT
Integration involves the systematic linking of system components, functions and solutions. For digital systems comprising low-cost components, integration enables the sharing of resources and allows them to benefit from each others functionalities. However, in general integrating low-cost systems can be difficult, since data is heterogeneous and distributed over multiple sources and there is no common interface to interact with resources. This study proposes the use of reference architectures as a means of supporting such integration needs. Reference architectures are models that provide a structured template with common terminology and support integration through a systematic, repeatable framework. In this paper, we evaluate strategies to integrate digital systems formed from the same reference architecture. We first select three reference architectures and use them to conceptually design the integration of two low-cost digital systems based on different strategies. Then, the conceptually designed strategies are evaluated in terms of their effectiveness to be realised at low cost. Finally, the most effective strategy is deployed for an industrially relevant application, and validated by comparing its effectiveness to a system architecture driven integration. The evaluation results suggest that coordinators with a common service bus and virtual data integration are effective for integrating low-cost digital systems.

KEYWORDS
System integration; low-cost; digital manufacturing; reference architecture

1. Introduction

As companies approach Industry 4.0, digital systems become increasingly interconnected to improve efficiency and productivity of manufacturing processes (Rojko 2017). For example, such digital systems help to connect product development with supply chains, which yields more adaptable manufacturing processes (Molina et al. 2005). One way to reach a high level of interconnectedness among digital systems is by integration. However, compared to larger companies, small and medium-sized enterprises (SMEs) require a different approach, since they tend to have a smaller budget, lack necessary technical skills and only have limited access to data (Schönfuß et al. 2021; Kaiser et al. 2021).

CONTACT J. Kaiser. Email: jk823@cam.ac.uk

The context of this study revolves around the integration of low-cost digital systems used in manufacturing. In particular, we focus on the most elementary form of integration which describes the combination of low-cost digital systems' functions and data. It is crucial to understand how this elementary form of integration can be achieved before more complex combinations can be designed.

A *low-cost digital system* is a special type of digital system, that consists of a combination of low-cost components and meets entry-level digitalisation needs of an SME (McFarlane et al. 2020). *Low-cost components* are inexpensive hardware or software elements that are part of a digital system. 'Low-cost' means that the cost of a digital system is only the fraction of the cost of a commercial product (Macias-Aguayo et al. 2022). Low-cost digital systems can be used to deliver a *solution* to a particular problem. The overall cost of a digital system is not governed by the components costs alone, but also includes the time spend for the design, development and deployment (Hawkridge et al. 2022). In this study, we are particularly interested in the cost (or effort) of integrating low-cost digital systems.

The low-cost digital systems considered here are typically non-safety-critical systems that are generally peripheral to core production processes. Common functions of such systems include sensing, decision support and operator guidance. Examples of low-cost digital systems include tracking and monitoring applications (Schönfuß et al. 2021). Cloud-based components can potentially be part of a low-cost digital solution since cloud-based service providers have become increasingly more affordable in recent years (Kaiser et al. 2021).

Integrating low-cost digital manufacturing systems and solutions enables them to share resources and benefit from each others functionalities. However, integration is difficult, as data is typically distributed over multiple heterogeneous sources and there is no common interface to interact with resources (Hasselbring 2000). To address these challenges, this study proposes using so called *reference architectures* to support the integration. Reference architectures provide structure in the design of systems, thus significantly reducing the effort required to develop and implement them (Kaiser, McFarlane, and Hawkridge 2022). While a large number of reference architectures have been proposed over the years, only few support the design of low-cost digital systems, and there are no explicit guidelines for the integration of systems that have been derived from the same reference architecture (Kaiser et al. 2023). In particular, this study seeks answers for the following research questions:

RQ1: What are the design challenges of low-cost digital manufacturing systems that are able to share data and resources?
RQ2: Which reference architectures are capable of supporting the development of low-cost digital manufacturing systems?
RQ3: What system and data integration strategies have the potential to support a low-cost digital manufacturing system integration?
RQ4: What strategy is most effective in integrating low-cost digital manufacturing systems?

This study proposes to use reference architectures as a means for supporting the integration of low-cost digital manufacturing systems. The objective of this study is twofold: (1) we conceptually design different integration strategies for three reference architectures, and evaluate to which extent they achieve an effective integration of low–cost digital systems. (2) We validate the rationale behind using reference architectures by comparing their effectiveness to a system architecture driven integration. While

this study concentrates on the integration of systems derived from the same reference architectures, there is need to examine the integration with external elements, which is subject to future study. In particular, this paper revolves around the most elementary form of integration by combining two low-cost digital systems. However, the developed reference architecture driven integration is also capable of forming more complex combinations and thus can form part of a broader integration, such as an entire factory supported by low-cost digital solutions. The key novelty of this work is that while the literature reports on the use of reference architectures in design and on integration challenges when combining digital systems, the use of reference architectures to guide integration has not received significant attention. In particular, the key contributions of this study include (a) analysing the capabilities of reference architectures to support a systematic integration of digital systems, and (b) evaluating the effectiveness of different reference architecture driven integration strategies for low-cost digital systems used in manufacturing.

The integration of systems including their resources and data has been studied extensively over the last decades. To avoid ambiguity, we begin by defining the concepts of system and data integration in the context of this study, and describe their differences.

1.1. *System integration*

System integration refers to *the sharing of resources and functionalities across multiple digital systems by following a specific integration strategy* (Kaiser et al. 2022a). Integrating multiple different distributed digital systems is beneficial because the functionality of an individual system might be enhanced by having access to the resources of another system, and capabilities can be shared between systems which saves time and effort during development. Within the context of this study, we identify *two types of system integration*, namely (a) integration of digital systems designed based on the same reference architecture, and (b) integration of such systems with external elements. These external elements can be legacy systems without any architectural description, parts of an existing IT infrastructure, or systems derived from a different reference architecture. This study only addresses the first type of integration and evaluates the effectiveness of different integration strategies for three different design approaches. Best practices of system integration can be generalised to architectural principles to facilitate an integration at low cost. Specifically, reference architectures should be modular, use abstract data types, and rely on explicit (open) interfaces (Nilsson, Nordhagen, and Oftedal 1990). A sufficiently high modularity of resulting systems is obtained via weakly coupled system components with little information exchange, that hide information of a component unless declared otherwise.

1.2. *Data integration*

Data integration describes *the process of connecting, managing and combining data from different sources.* While system integration involves the sharing of data between digital systems, data integration provides a unified view to enable an easy access and consumption of data. Indeed, the different data formats and non-aligned structures are a key challenge when integrating data of distributed digital systems (Modoni et al. 2017). The majority of data integration strategies fall on the spectrum between *data warehousing* and *virtual integration* (Doan, Halevy, and Ives 2012). Data warehousing

loads data from individual sources and incorporates the data into a single storage. Extract-transform-load (ETL) pipelines are used to periodically extract data from the sources to the warehouse, which consists of an underlying database and a logical schema which relates the data from the individual sources to the warehouse. On the other hand, for virtual integration, data remain in their sources and are accessed at query time. The warehouse is replaced by a mediated schema which enables the posing of queries, and wrappers are used instead of ETL pipelines to handle the communication with the data sources. Apart from that, there are a number of characteristics that need to be considered when integrating data of low-cost digital systems (Patel 2019; Schönfuß et al. 2021; Kaiser et al. 2021): while the data is likely to be static, structured and has a low volume, their data sources tend to be heterogeneous. Additionally, the applied integration strategy should not put additional load on the individual data sources, and should not affect the performance of the individual systems or the integrated whole.

1.3. *Paper outline*

The paper is structured as follows. Section 2 provides an overview of integration strategies and architectures for low-cost digital systems, and outlines key design challenges of low-cost digital systems. Section 3 presents the methodology. The following two sections constitute the results of the work. Section 4 conceptually designs and evaluates selected integration strategies, while Section 5 deploys and validates the most effective integration strategy. Section 6 discusses the results of this study. We conclude this paper by proposing future research directions.

2. Background

Integration constitutes a key issue in the design of digital systems. Much work has been done on providing structure for integrating systems, especially in the software engineering domain. However, one of the limitations of this work includes analysing the capabilities of reference architectures to support an integration at low cost, and evaluating the effectiveness of different integration strategies. This section outlines research gaps in the study of system integration and establishes the focus of this paper. We begin by reviewing existing integration approaches and architectures for low-cost digital manufacturing, which is followed by a list of key design challenges of low-cost digital systems.

2.1. *Systematic approaches to system integration*

There are numerous systematic approaches to integrate digital systems, most of them concentrate on the integration of disparate systems. As an early approach to integration within a factory is developed by Jones and McLean (1986) who propose a generic template upon which to base system designs and interfaces. Interoperability is the main focus of Nilsson, Nordhagen, and Oftedal (1990) who propose to use a *centralised data storage* and a mechanism to enable data transfer, for example, via software *agents* or a *service bus*. The authors state that since a centralised data storage may not yield sufficiently open systems, developers should utilise data abstractions and extend data with semantics to increase the capabilities of the integrated system.

A review of commonly used integration patterns is provided by Mularz (1995). The main integration patterns are *message brokers*, wrappers, work flow managers and *central data storage*. Message brokers and wrappers are required to provide continued access to legacy systems. Alternatively, work flow managers can be employed to automatically perform user-defined tasks based on the execution of stand-alone components. This could be implemented through scripts that control the execution context. Moreover, a less constrained approach to messaging can be achieved through brokers, which use services to communicate between independently operating applications. To share information among systems a central data repository is needed. Land and Crnkovic (2003) propose four approaches to integrate systems: first, to integrate systems into enterprise applications, interface wrappers and adapters for each system are required. Second, manually importing and exporting data might enhance interoperability, but data inconsistencies are likely and systems may become incapable of acting autonomously. Third, when data is shared via a *centralised database*, existing systems must be modified to use this central repository. Fourth, systems are integrated by merging source code. The authors further describe a fast low-risk approach to integration that consists of starting with manual integration and successively replacing data imports and exports with communicating elements and integrated repositories. For software legacy systems, Land and Crnkovic suggest to rearrange the architectural description to meet the requirements of the application that the legacy system needs to integrate with. Hepner et al. (2006) identify three main elements that are part of any system integration approach, specifically extenders, translators and controllers. Extenders add features to enable components to complete their message transfer. Translators carry out data transformations and mappings. Controllers *coordinate* the information exchange among systems and their components. Further, Sabooniha, Toohey, and Lee (2012) distinguish between different solutions for the integration of information systems in healthcare. The main approaches include using databases with a standardised messages exchange, a *central application layer* on top of existing digital systems, manual synchronisation/*coordination* of systems, and employing services to share functionalities and data. Zhang et al. (2015) propose a framework to integrate real-time manufacturing information with management systems. Beregi et al. (2021) develop a service-oriented Manufacturing Execution System (MES) to integrate and *coordinate* different cyber-physical production systems (CPPS). Besides relying on a specific integration strategy, the use of common interoperability standards gives potential to facilitate the process of integrating digital systems. In the manufacturing domain, OPC UA (IEC TR 62541-1 2020) is a widely used approach based on a service-oriented architecture. OPC UA defines a platform-independent machine-to-machine communication protocol for industrial automation. Many enterprise-level reference architectures recommend using this standard for supporting the integration of digital systems into a larger application (Kaiser et al. 2023).

In addition to the above, a handful of studies deal with integrating digital systems in specific scenarios: Covanich et al. (2007) integrate a machine into an existing digital manufacturing system by wrapping resources into holons. Cruz-Correia (2010) summarises lessons learned from integrating hospital information systems. In particular, the author suggests that user interfaces and databases should adapt automatically and agents should be used for complex heterogeneous environments. Moreover, Chen, Chen, and Hsu (2014) combine Internet of Things (IoT) and services to integrate physical resources for a pallet inventory tracking application. Finally, Fang et al. (2015) leverage IoT and cloud services to create an integrated digital system for a snowmelt flood early-warning system. It is noteworthy to mention that the aforementioned system in-

tegration approaches differ from Enterprise Application Integration (EAI) (Gudivada and Nandigam 2005; Yang and Lu 2005). EAI is concerned with integrating systems and services an enterprise uses on a conceptual level, which is beyond the scope of the low-cost digital systems considered in this paper.

2.2. *Low-cost data integration approaches*

Although the majority of system integration approaches propose a shared data storage, attempts to centralise data in terms of data warehousing or virtual integration are rare. Patel (2019) outlines the benefits of using messaging brokers and a cloud-based storage for data integration. The author describes three general steps to integrate multiple data sources, namely a discovery mechanism to identify information that is beneficial for other systems, data extraction via schemas, and data export to a temporary storage or a messaging system. Hasani, Sadeghi-Niaraki, and Jelokhani-Niaraki (2015) propose an ontology-based approach for highly heterogeneous sources of spatial data, which consists of creating and mapping relative ontologies for each database.

There are a number of studies that aim to integrate data from different sources at low cost. They cover the manual import of data into a single database (Wilamowska et al. 2013; Mahmoodpour and Lobov 2019), using a remote script to combine data from different sources (Coronado et al. 2018), and facilitating data warehousing by using a graphical tool (Hilario et al. 2021) or limiting the structure and variety of data (Sahara and Aamer 2021). Apart from that, methods of integrating data of mechanical parts can also be used for integrating data of digital systems in a low-cost manner, for example, by converting data to a standard format (Mostefai, Bouras, and Batouche 2005) or using a data model that includes the semantics shared between digital systems (Feng and Yang 1995).

A key issue when integrating data from separate heterogeneous sources is that each source typically uses different data formats and interoperability standards. Much work has been done to overcome this issue, in particular, through the use of common semantic models. For example, Modoni et al. (2019) propose to use a semantic digital twin as part of an event-driven framework to represent the resources within a factory. This semantic model is shared by all distributed resources connected to that framework, thus providing a common vocabulary adopted by those resources. Apart from that, Müller et al. (2022) utilise a uniform information model to make the information of CPPS accessible to other applications.

2.3. *Reference architectures and system architectures for low-cost digital systems*

Industrial digital systems can be designed in a number of ways. A common design approach describe the use of architectures. There are two main types of architectures relevant for this study, namely reference architectures and system architectures. *Reference architectures are models that provide a structured template with common terminology* (Kaiser et al. 2023). These general templates guide developers to design specific system architectures and inferred digital systems. Although a large number of reference architectures have been proposed over the years, only few provide sufficient support for the design of low-cost digital systems and there are no explicit guidelines for their integration. In particular, these reference architectures are *modular*, since modularity becomes necessary to balance the cognitive load when considering the large

variety of components and relations in those systems, and they have a *low level of abstraction* by providing detailed implementations guides, technology and standards recommendations, and unified interfaces (Hawkridge et al. 2022). We identify three reference architectures relevant to low-cost digital systems, namely Shoestring, WoT and PROSA/Erlang (Kaiser et al. 2023):

Shoestring. The so called *Shoestring* approach (McFarlane et al. 2020) aims to enhance the digital capabilities of SMEs by providing a design method that enables the use of low-cost, readily-available, off-the-shelf components. Its reference architecture provides a modular solution template with *service modules* and *building blocks* (Hawkridge et al. 2022). Building blocks are modular components with elementary functionalities and well-defined interfaces, which encapsulate components, such as Raspberry Pis or sensors (Kaiser et al. 2022b). Service modules combine multiple building blocks using wrappers communicating via a service bus, and provide high level functionalities at a network level, such as data collection and data storage.

WoT. The *Web of Things (WoT)* (Lagally et al. 2021) aspires to enable interoperability across IoT system domains by preserving and complementing existing solutions and standards. Its reference architecture abstracts any physical or virtual entity and exposes it as a web resource. Based on the concept of web *Things*, resources are described through standardised metadata and *consumers*, which process this data and interact with resources. For manufacturing, Things can be sensors or a web page, while consumers are typically computational devices, such as Raspberry Pis. While WoT does not prescribe any specific cooperation mechanisms, Things and consumers typically interact via internet application protocols. This study uses the official reference implementation of WoT[1].

PROSA/Erlang. The *PROSA* (Van Brussel et al. 1998) reference architecture uses autonomous cooperating agents, so called *holons*, to represent the roles and components of a manufacturing system and enhance its fault-tolerance: *Product holons* hold the process and product knowledge, *resource holons* contain a physical part of the manufacturing system and an information processing part which controls the resource, and *order holons* characterise a task in the manufacturing system and manage the production of the physical product. The order and resource holons exchange information regarding the execution progress of operations on resources. Since PROSA is more abstract than the other two due to a lack of implementation guides, we use a specification based on Erlang/OTP[2] (Kruger and Basson 2017).

While Shoestring, WoT and PROSA/Erlang differ in their concept and implementation, their key architectural elements are similar, which yields comparability. Figure 1 exemplifies 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 components, and provides their functionalities and data at a network level.

More broadly, beyond these three specific reference architectures for digital systems, there are a wide range of other reference architectures that can be used to support digital manufacturing system design and integration challenges (Kaiser et al. 2023). In particular, the ISA-95 (IEC 62264-1 2003) standard and the Reference Architecture Model Industry 4.0 (RAMI 4.0) (IEC/PAS 63088 2017) are most commonly known for supporting the integration of digital systems used in manufacturing. ISA-95 integrates

[1] `https://github.com/eclipse/thingweb.node-wot`

[2] `https://www.erlang.org/docs`

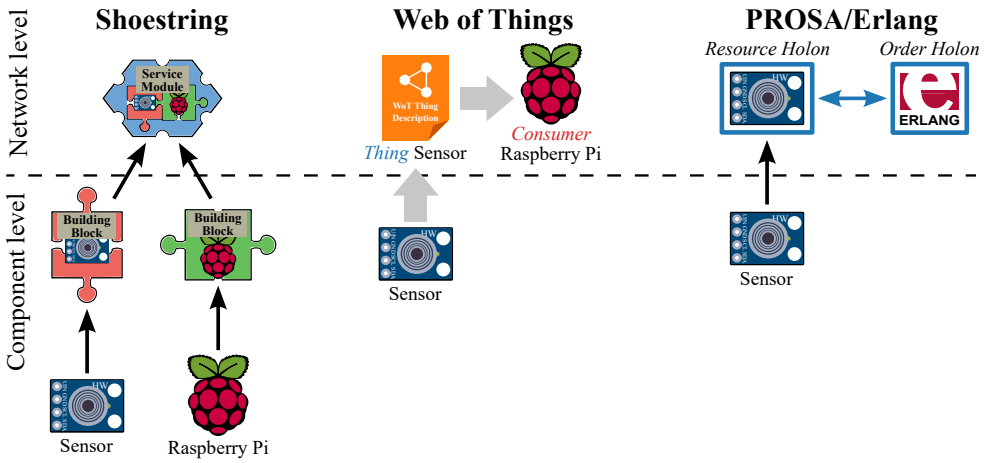


Figure 1. Example of how Shoestring, WoT and PROSA/Erlang encapsulate system components into their respective architecture elements (service modules, Things, holons), which provide data and key functionalities at a network level and thus fulfill equivalent roles upon comparison

control and enterprise systems through a hierarchical model of production and business processes including components of the control and information system. RAMI 4.0 relies on standards and specifies life cycle, technical and organisational functions of system components. These components (or assets), such as machines, are encapsulated into an administration shell, which offers a standardised interface and data storage capabilities. However, as shown by Kaiser et al. (2023), ISA-95 and RAMI 4.0 are oriented at the enterprise level and provide fewer specific implementation details for individual digital systems and require designers to make many additional (undirected) decisions during the design process compared to the three reference architectures mentioned above. We have thus selected Shoestring, WoT and PROSA/Erlang, since they are likely to require the least decision making effort when designing and integrating low-cost digital systems. In contrast to ISA-95 and RAMI 4.0, these three reference architectures are more applicable due to their detailed design guidelines, submodels and standard recommendations, and their narrow range of development pathways, which results in a more straightforward design process.

System architectures conceptualise the structure and function of an implemented digital system via virtual descriptions of its assets and interactions (Kaiser et al. 2023). While reference architectures generally describe all system elements and their interactions that could be considered for different applications, system architectures are composed of a subset of these elements selected for a specific application. System architectures integrate systems in a less structured ad-hoc manner. However, they could potentially require less integration effort than a reference architecture. Although there are no system architectures that explicitly focus on low-cost digital systems, several include features that facilitate the integration of such systems. In this study we focus on the BaSyx[3] system architecture since it incorporates various open-source tools and detailed implementation guidelines, which makes it particularly well-suited for a low-cost integration.

BaSyx. BaSyx is a middleware based on RAMI 4.0 (IEC/PAS 63088 2017) to support Industry 4.0 production environments (Schnicke et al. 2022a,b). It connects production assets through a virtual bus and implements all components necessary for an end-to-end digitalisation (Kuhn, Antonino, and Schnicke 2020). Assets are virtualised

[3]`https://github.com/eclipse-basyx/basyx-java-sdk`

through *Asset Administration Shells (AAS)*, which contain data representations and control components. Data is represented by *submodels* which provide different views on the asset. For example, a first submodel could provide access to the database of a system, while a second one collects sensor data. AAS provide a unified communication interface for the middleware. Key components of BaSyx include the AAS server, which provides access to the shells and submodels, and a registry tracking their availability.

BaSyx characterises a system architecture because it conceptualises the structure and function of the implemented system via virtual descriptions. It details the components of the system by means of assets and AAS, and their interactions, which are managed by shell managers. BaSyx is more abstract compared to other system architectures in the literature (Kaiser et al. 2023), because it allows for any type of production environment asset to be encapsulated in an AAS, thus requiring more design decisions. However, the number and range of these decisions are limited by the architectural components, and hence, BaSyx is more concrete than Shoestring, WoT and PROSA/Erlang.

2.4. *Low-cost digital system design challenges*

This subsection outlines the key design challenges that are associated with low-cost digital systems, since these have a significant impact on the effectiveness of a particular integration strategy. In the literature a number of themes have been identified that revolve around reusable components, standardised interfaces and open-source technologies. For example, a low-cost digital system *should be* modelled as a sequence of *reusable* objects that combine both functions and data (Roşca, Bănică, and Sîrbu 2010) since reusing components from previous systems reduces the effort during the development of new systems. Moreover, Filip (2011) argues that selected information and communication *technologies need to be affordable* and *should easily integrate with legacy systems* by adopting an open system architecture, standardised data formats, and semantic unification. Selecting appropriate connectivity standards and technologies may facilitate the integration of disparate systems (Hawkridge et al. 2019). However, since the low-cost digital systems derived from the reference architectures are modular, service-oriented standards, such as OPC UA and MTConnect, are likely to be required. As part of several non-technical challenges, low-cost digital systems *should be easily accessible to non-expert users* (Schönfuß et al. 2021), for instance, by minimising the number of functions. For digital manufacturing systems, *hardware and software should be open-source* while developers should aim to *achieve acceptable levels of reliability and safety* (Hoxha et al. 2016). Additionally, low-cost digital systems are likely to be created by *joining disparate components* via standardised interfaces and such systems may be *developed successively* instead of as an integrated whole (Erbe 2002; McFarlane et al. 2020).

2.5. *Research gaps*

There has been a lot of work on integrating systems and data. However, there are a number of limitations of this work to date, which we aim to address in this study: first, since the majority of studies focus on the integration of software systems, more work needs be done on studying the integration of digital systems that contain both software and hardware components. Additionally, limited work has been done on the integration of particularly low-cost systems. Second, while most studies propose multiple integra-

tion strategies, there is a need for comparing and evaluating the effectiveness of these different strategies. Third, current integration strategies lack an underlying structured design approach. There is a need for analysing the capabilities of an architectural design approach to support the integration of digital systems.

3. Methodology

This section outlines the methodology used in this study. In particular, we propose criteria of an effective integration of systems and data at low cost, and conceptually design integration strategies based on three different reference architectures. Thereafter, the proposed criteria are applied to evaluate the conceptual designs of these integration strategies in terms of their effectiveness to be implemented at low cost. Then, trial deployments of the most effective integration strategy are developed for an industrially relevant application. Finally, this integration strategy is validated by evaluating the effectiveness of a system architecture driven integration and comparing it to these trial deployments. Specifically, the methodology used here consists of five steps:

1. Define criteria of an effective integration of digital systems and data at low cost
2. Conceptually design the integration of two low-cost digital systems based on different strategies and reference architectures
3. Evaluate the conceptual designs of these strategies in terms of their effectiveness to be realised at low cost
4. Develop trial deployments of the most effective integration strategy based on the different reference architectures for an industrially relevant application
5. Validate the rationale behind using reference architectures by evaluating the effectiveness of a system architecture driven integration and comparing it to these trial deployments

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

3.1. *Definition of criteria of a low-cost integration*

We argue that integrated low-cost digital systems have similar design challenges as individual systems because *integration can be considered as an aggregation of individual systems*. In other words, the integrated system consists of a set of individual systems which in turn form a larger system with its own identity. The individual systems can be seen as components of the integrated whole. The concept of aggregation stems from object-oriented programming, but has also been applied to other areas, such as holonic manufacturing systems (Van Brussel et al. 1998). It is noteworthy to mention that in some cases an integrated digital system can provide significantly more value than the simple aggregation of the outputs of individual systems. For example, a predictive maintenance solution that analyses the data of single sensor could be enhanced by combining it with the sensor data of other systems monitoring similar machines, since the combined data is likely to yield more precise predictions. Although the study in this paper is mainly concerned with the most elementary form of integration - simply combining and connecting low-cost digital manufacturing systems - it is worth noting the significant potential gains from effective integration. Based on design challenges described in Section 2.4, we define a set of criteria that a particular strategy should satisfy to be considered low-cost.

3.2. *Selection of integration strategies for the conceptual designs*

There are a number of different strategies to integrate systems and data. Some of them have the potential to be low cost. However, the proposed strategies only address specific features of an integration, such as data management in terms of data storage centralisation or the coordination of resources. Not all of these integration features are mutually exclusive: often performing an integration based on one strategy entails incorporating characteristics of another strategy. For example, a central layer managing multiple applications always requires some type of communication mechanism. Therefore, there is a need for a complete integration approach. We identify five integration strategies that potentially require little effort when integrating digital systems derived from the same reference architecture: (1) *Data warehousing* and (2) *Virtual integration* as part of data centralisation, (3) *Coordinator-based integration*, (4) *Integration via a common service bus*, and (5) *Integration via a central application layer*. To ensure that all aspects of the integrated digital systems are captured, the recommended most effective integration strategy may be a combination of multiple strategies. To identify this recommended strategy, an integration of two low-cost digital systems based on the five selected strategies is conceptually designed.

3.3. *Evaluation of conceptual designs*

The conceptual designs of the different integration strategies are evaluated to identify a recommended strategy which is most effective in integrating digital systems at low cost. In particular, we apply the criteria of a low-cost integration to the conceptual designs of each strategy for an integration based on three reference architectures, namely Shoestring, WoT and PROSA/Erlang. The three reference architectures are exemplary design approaches, which help to verify the evaluation regarding an effective integration of low-cost digital systems.

3.4. *Development of trial deployments for an industrially relevant application*

For validating the the rationale behind using reference architectures, it is necessary to implement the recommended integration strategy and compare these trial deployments to an integration based on a system architecture. Since system architectures generally impose more strict design guidelines than a reference architecture, a conceptual design and comparison is not feasible. The trial deployments are developed for Shoestring, WoT and PROSA/Erlang by explicitly applying the recommended strategy to an industrially relevant use case. Specifically, we combine two different low-cost digital systems to solve the needs for a timber doorset manufacturer in the UK.

3.5. *Validation of the reference architecture driven integration*

The validation of the reference architecture driven integration entails comparing its effectiveness to a system architecture driven integration. Since system architectures do not allow for a conceptual design and evaluation of different integration strategies due to their strict design guidelines, we evaluate a deployment of a system architecture driven integration. Specifically, we implement an integration based on the BaSyx system architecture and evaluate its effectiveness by applying the criteria of a low-cost

integration. We then compare the integration based on BaSyx to the trial deployments based on Shoestring, WoT and PROSA/Erlang.

In the following two sections, the proposed methodology is carried out. The first section captures the first three steps to identify the most effective strategy for a reference architecture driven integration, while the second one develops the trial deployments and performs the validation.

4. Conceptual design and evaluation of integration strategies

This section captures the first three steps of the methodology. First, criteria for a low-cost integration are defined. Then, the selected strategies and reference architectures are used to conceptually design the integration of two low-cost digital systems. Finally, these conceptual designs are evaluated by applying the criteria to identify a recommended strategy for a reference architecture driven integration.

4.1. *Low-cost integration success criteria*

As mentioned earlier, an integrated digital system is an aggregation of individual digital systems. Hence, we argue that integrated digital systems are subject to the same design challenges as individual systems, which are described in Section 2.4. Based on these challenges, we define the following criteria that a particular strategy should meet to be considered low-cost for an integration of digital systems derived from the same reference architecture:

- **Facile** - Integrating system components (including data, resources and functions) should preferably occur at a network level. By default, information relevant to other components of the same system, or another system, should be made available at a network level via architectural descriptions. If information is not relevant for other components within an individual system, it is not considered relevant for other systems. Besides, developers should leverage the well-defined interfaces of architectural components at the network level, since less effort is required to form connections through these interfaces than would be needed if low-level component interfaces were used. However, if a reference architecture only provides functional capabilities, integrating systems is expected to be more laborious.
- **Successive** - The integration strategy should promote a step-by-step incorporation of additional systems and components. The system boundaries should not be rigid; extra architectural components may be added in order to increase the capabilities of the main system, before transitioning to multiple integrated systems.
- **Scalable** - It is likely that systems will be required to evolve over time with the introduction of new production requirements. For example, additional features may be needed, such as another sensor location for a monitoring system, or the integration with another system. Therefore, the integration strategy should be able to integrate numerous systems through an appropriate interaction scheme.
- **Seamless** - As production requirements may change over time, only little development effort should be required to adapt the integrated system to those changes. Hence, systems and components should be able to be integrated and

disintegrated seamlessly, whereby 'seamless' implies a low number of development steps and system modifications required to conduct the integration or disintegration. Likewise, redundancies in terms of resources and data should be minimised.

- **Reliable** - Although essential to any type of manufacturing systems, reliability is particularly important for integrated systems, since the failure of one system may affect the operation of other systems. Thus, integrated digital systems and their components should perform to an acceptable level in a manufacturing setting.
- **Transparent** - The integration strategy should clearly indicate the connections and interactions among systems and components. Transparency is only considered in terms of configuring the system interactions. From an operation perspective, an individual component is unaware of all components it can interact with due to the components' information hiding.

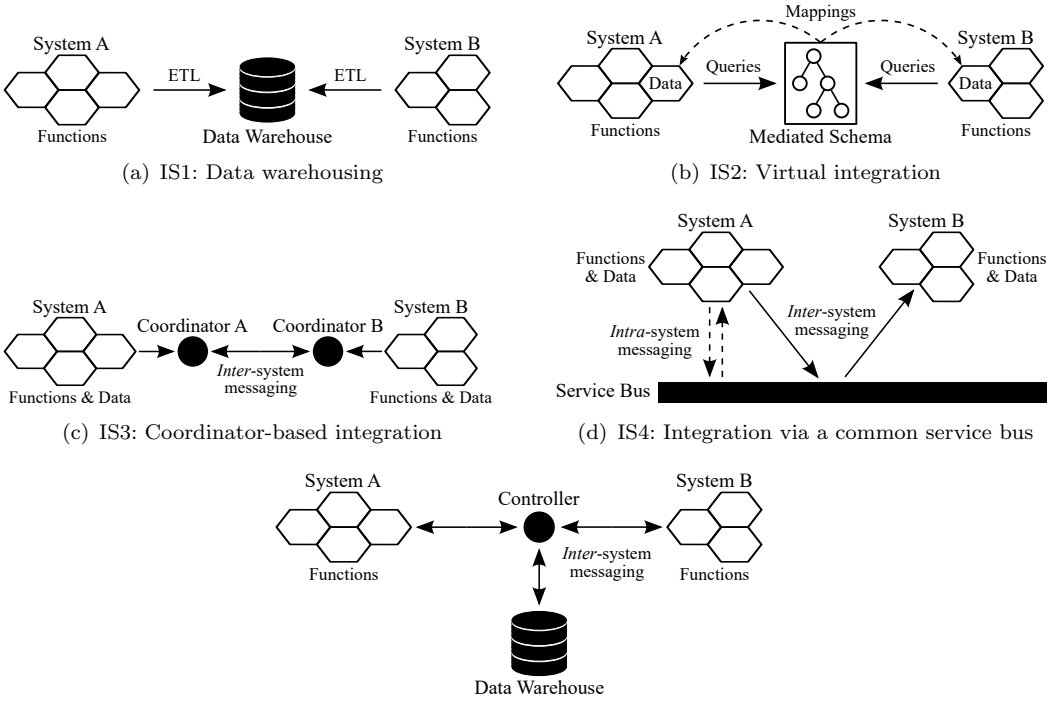
4.2. *Conceptual design of integration strategies*

In this section, we are going to evaluate a number of integration approaches that have the potential to be realised at low cost. To identify an effective low-cost integration strategy, the conceptual designs of each strategy (IS1-5) is evaluated by applying the aforementioned criteria. In what follows, the conceptual design for each strategy, which is shown in Figures 2(a) to 2(e), is described for an integration of two general low-cost digital systems that have been derived from one of the three modular reference architectures. Shoestring, WoT and PROSA/Erlang are exemplary design approaches, which help to verify the evaluation regarding an effective integration of low-cost digital systems.

IS1: Data warehousing. Data warehousing, as shown in Figure 2(a), is a strictly hierarchical approach that stores data of multiple digital systems in a single resource. This resource can be physical or virtual, such as a cloud-based storage. Data is shared among those systems using ETL pipelines. While data warehousing may reduce data redundancies, distributed data storages have a higher responsiveness for live data and enable an easy integration of new systems due to their modular structure.

IS2: Virtual integration. The data storages of systems remain separated and data is shared via a mediated schema, which is shown in Figure 2(b). In this case, wrappers are not required since the key architectural elements that operate at a network level provide standardised interfaces. For example, one system can access the data of another system by posing queries to a logical schema that links to the data source of the other system.

IS3: Coordinator-based integration. The main objective of system integration is to identify and link data streams and sources among multiple systems. Coordinators are a hybrid approach that organise connections among systems through an element central to each individual system, which remains inactive for a single system. This central element can take three different roles, namely passive, active or a Single Source Of Truth (SSOT). In passive coordination, the connections between system components are configured manually and coordinators simply map these connections onto the underlying system. Active coordinators control the system by steering the data flow. They may act as a broker or gateway, channelling all data flows through themselves,



(a) IS1: Data warehousing

(b) IS2: Virtual integration

(c) IS3: Coordinator-based integration

(d) IS4: Integration via a common service bus

(e) IS5: Integration via a central application layer

Figure 2. Conceptual design of different integration strategies (IS1-5). Each hexagon represents an architecture element that makes a subset of the data and functions of the low-cost digital system available at a network level.

or take up the role of a switchboard, that manages direct communication between resources. For example, active coordinators can be implemented via software agents. In an SSOT architecture, every system component is mastered in only one place, which is discussed in more detail for the central application layer. Figure 2(c) illustrates the conceptual design of coordinator-based system integration.

IS4: Integration via a common service bus. As part of a service-oriented architecture (SOA), a service bus is a heterarchical approach and offers a common interface for communicating with systems and components. For example, a common service bus can be built based on the publish-subscribe pattern of MQTT. If a reference architecture is service-oriented, its service bus can be expanded to include messages sent by other systems, which is shown in Figure 2(d).

IS5: Integration via a central application layer. A central application layer as shown in Figure 2(e) is a central controller for managing and distributing functionalities and data among systems, thus creating a Single Source Of Truth (SSOT) for the integrated systems. It consists of a combination of different integration strategies: first, a data warehouse stores the data of each individual system. Links to other components are by reference only. Second, a central coordinator governs the functions and data of the separate systems. Third, a common service bus is used to handle intra and inter-system messages. A central application layer could be implemented via a software agent that coordinates resources of multiple systems and manages their data in a database.

Since each strategy has the potential to support the integration of low-cost digital manufacturing systems, there is a need for evaluating their effectiveness for integrating such systems to identify a recommended strategy when using a reference architecture.

4.3. *Evaluation of the conceptually designed integration strategies*

To identify a recommended strategy which is most effective in integrating digital systems at low cost, the conceptual designs of the selected integration strategies are evaluated by applying the criteria of a low-cost integration. Since there are conceptual differences between Shoestring, WoT and PROSA/Erlang, it is necessary to evaluate each reference architecture individually. The evaluation results are summarised in Table 1.

Facile. Data warehousing requires ETL pipelines and a schema to map data to each system, which is a modification below the network level. Hence, data warehousing and a central application layer are not facile for Shoestring and WoT. However, data warehousing is less laborious for PROSA/Erlang designs since individual data storage holons can be aggregated to a logical warehouse, which serves as a gateway to the data of each system. Virtual integration, coordinators and a common service bus can be implemented without modifying system components, which yields a lightweight integration. Specifically, queries to the mediated schema can be posed via the existing network level interface. However, the elements of WoT already provide capabilities for semantically linking data. Therefore, developing an additional mediated schema creates an extra layer of complexity and thus should be avoided.

Successive. Data warehousing does not provide the capabilities for a low-cost step-by-step integration of systems for Shoestring, WoT and PROSA/Erlang, because the developer needs to physically integrate the data sources when a new system or component is introduced. In contrast, little effort is required for virtual integration since only the mediated schema needs to be updated. Further, a coordinator-based integration is likely to be achieved at low-cost for the three reference architectures, since coordinators can make use of the same network as the other architectural elements. Such networks can be implemented via a common service bus, which enables a flexible extension of resources. However, while coordinators organise the connections among systems and components, a service bus requires fewer overall components, and thus, the design effort is lower. For a central application layer, the main controller could simplify the physical integration of distributed data storages through a common scheme or an ontology, which leads to a more successive integration for Shoestring and WoT. However, for PROSA/Erlang these functions are handled by the order holons and therefore using a common scheme or ontology would yield additional complexity.

Scalable. For the three reference architectures, data warehousing may become scalable with an appropriate storage model. Nonetheless, a central application layer becomes a bottleneck for Shoestring and WoT as the number of systems increases, because the main controller needs to be modified whenever a new system is added, and it may also require a prioritisation scheme. PROSA/Erlang can delegate coordinating tasks to order holons, thus yielding scalable systems. Virtual integration is highly scalable for Shoestring, WoT and PROSA/Erlang as the data sources remain physically separated. In contrast, the message broker of a service bus may limit the scalabil-

Table 1. Evaluation of the effectiveness of the five proposed integration strategies

Shoestring	Facile	Successive	Scalable	Seamless	Reliable	Transparent
Data warehousing	-	-	0	-	-	+
Virtual integration	+	+	+	+	+	+
Coordinator-based	+	+	+	+	+	0
Common service bus	+	+	0	+	+	-
Central application layer	-	0	-	-	-	+
WoT	Facile	Successive	Scalable	Seamless	Reliable	Transparent
Data warehousing	-	-	0	-	-	+
Virtual integration	0	+	+	0	+	+
Coordinator-based	+	+	+	+	+	+
Common service bus	+	+	+	+	+	-
Central application layer	-	0	-	-	-	+
PROSA/Erlang	Facile	Successive	Scalable	Seamless	Reliable	Transparent
Data warehousing	0	-	0	0	-	+
Virtual integration	+	+	+	+	+	+
Coordinator-based	+	+	+	+	0	+
Common service bus	+	+	0	+	+	-
Central application layer	0	-	0	0	-	+

ity of the integration approach for Shoestring and PROSA/Erlang. However, latency only comes into effect for a large number of MQTT subscribers Collina, Corazza, and Vanelli-Coralli (2012) and Erlang allows for a large number of concurrent processes. For WoT, large complex environments are likely to consist of multiple servers exposing resources, which distribute the load among the computational devices and thus lead to a high scalability. Finally, coordinators can be easily extended as Shoestring, WoT and PROSA/Erlang are highly distributed by nature.

Seamless. Despite data redundancies can be avoided, centralisation by means of warehousing or a main controller is not seamless, because these elements cannot be disintegrated easily for Shoestring and WoT. Conversely, the PROSA/Erlang elements may aggregate to a form a logical warehouse, which combines the functionalities of individual data storage resource holons, thus allowing for a more seamless integration and disintegration. Moreover, coordinators, a common service bus and virtual data integration are highly distributed, and consequently, little development effort is required to add or remove systems and components for the three reference architectures. As mentioned above, since WoT already provides capabilities for linking data, an extra development step is required to construct the mediated schema on top of these existing features.

Reliable. Data warehousing and a central application layer turn into a single point of failure for an integrated system based on the three reference architectures. To reach acceptable levels of reliability in an industrial environment, additional layers of security, such as resource redundancies, are needed. On the contrary, coordinators, a service bus and virtual integration decouple systems including their data and functionalities, thus increasing their fault-tolerance for Shoestring and WoT. If a system or component fails, the operation of the remaining systems is not affected. For PROSA/Erlang, order holons can either represent a gateway between resources of an individual and other systems, or portray a task or operation of a system. For the former, the order holon is at risk of leading to a failure of the system it manages, if the operation of this holon is disrupted. For the latter, a new order holon can simply be generated, if a task fails to complete, which would yield a more fault-tolerant system.

Transparent. Centralisation and virtual integration yield a high transparency for the three reference architectures, since explicit data mappings are created. For instance, the logical schema of a warehouse semantically links to the integrated data, which are tailored to each system that soughts to extract that data. For WoT and PROSA/Erlang, coordinators indicate which systems and components are connected to and interact with one another, thus yielding a high degree of transparency. Conversely, Shoestring coordinators are unaware of the identity of other coordinators, which reduces the transparency of the integrated system. Additionally, a common service bus is often based on a publish-subscribe pattern, where elements publishing data do not know those receiving the data. Therefore, a common service bus causes a notably low transparency for the three reference architectures.

In summary, despite their conceptual differences, the three considered reference architectures should use the same integration strategy. Specifically, *coordinators in combination with a common service bus and virtual data integration* is likely to be achieved at low cost when integrating digital systems that have been derived from Shoestring, WoT or PROSA/Erlang.

5. Deployment and validation

Having decided that the most effective integration strategy is based on coordinators, a common service bus and virtual data integration, this section captures the last two steps of the methodology. First, the recommended integration strategy is used to combine two specific low-cost digital solutions based on Shoestring, WoT and PROSA/Erlang for an industrially relevant application. In this section, we refer to digital systems under development as 'solutions' because they are assembled to solve a need for an SME. The solutions in each case are Job Tracking and Digital Job Cards. Second, the proposed reference architecture driven integration is validated by assessing its effectiveness compared to a system architecture driven integration. Specifically, an integration based on the BaSyx system architecture is developed and compared to the three reference architecture trial deployments.

5.1. *Trial deployments for an industrially relevant application*

A timber doorset manufacturer in the UK deploys two low-cost digital systems, namely Job Tracking and Digital Job Cards. Job Tracking stores the locations of current jobs, captures events of newly started jobs and provides this information via a dashboard. Digital Job Cards provides operators at workstations with information about specific work orders, such as the type and quantity of a product that needs to be manufactured. Both solutions use barcode scanners to receive information about an item. As a result from an in-company requirements workshop, the manufacturer requires the combination of both systems, such that they use the same barcode scanner, which would simplify the operation and reduce procurement costs. Therefore, the recommended strategy is explicitly applied to combine Job Tracking with Digital Job Cards, which results in three trial deployments, one for each reference architecture. The three deployments of the integrated system are shown in Figures 3(a) to 3(c). Despite the same integration strategy, the three deployments differ slightly upon comparison due to the fact that the reference architectures vary in their concept and implementation.

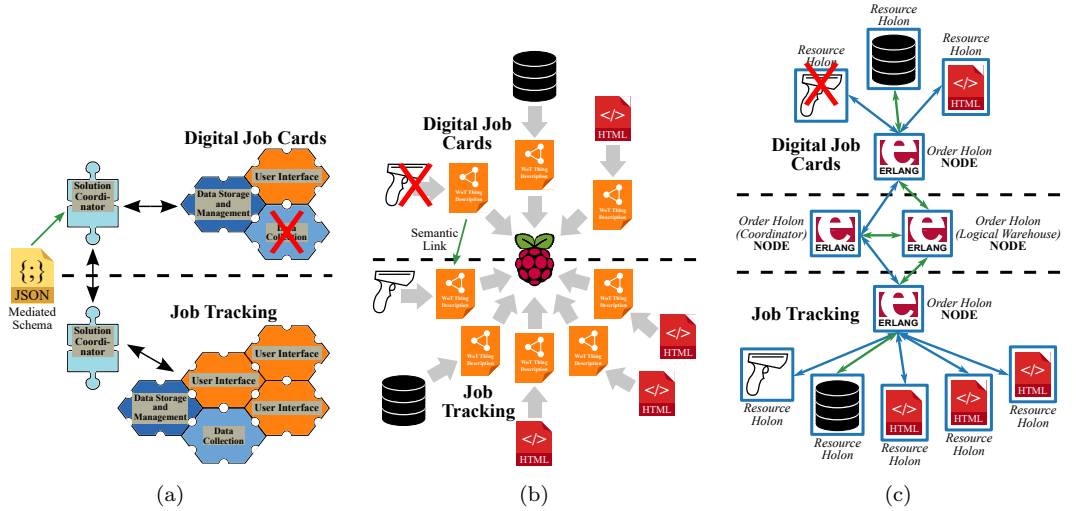


Figure 3. The combination of Job Tracking with Digital Job Cards for the timber doorset manufacturer based on (a) Shoestring, (b) WoT and (c) PROSA/Erlang. This figure only depicts the key relations between architecture elements and components. The dataflow among these elements is bi-directional.

For Shoestring, all messages are managed by a single MQTT broker, which serves as a common service bus. The coordinators are a variant of the service wrappers and use a mediated schema to translate the messages from one system to the other. Specifically, the barcode scan of Job Tracking is republished on a different topic.

For WoT, the coordination of messages is handled by the same consumer. All Things are exposed on the same network, which serves as the common service bus. Instead of a global mediated schema, a semantic link is used to share the barcode scan among the systems to leverage the in-built capabilities of WoT.

For PROSA/Erlang, the order holons merely serve as coordinators and do not represent a task or operation of the digital system. Specifically, a logical warehouse is constructed, which consists of an order holon as a gateway to the databases of the individual systems. Additionally, an order holon managing the inter-solution messages is implemented to minimise the source code modifications of the systems' order holons. Each order holon occupies a separate Erlang node to increase the fault-tolerance of the integrated system.

5.2. *Comparing system architecture with reference architecture driven integration*

System architectures provide a structured representation of *individual* systems or deployments. Hence, due to their low level of abstraction, system architectures might potentially provide more concrete support when integrating systems compared to the support provided by reference architectures. To explore this, we analyse the effectiveness of a system architecture driven integration. Since a conceptual design and evaluation of different integration strategies is not feasible due to the strict design guidelines of system architectures, it is necessary to perform the validation through a comparison of deployments. Specifically, we apply the criteria defined in Section 4.1 to evaluate the integration of Job Tracking and Digital Job Cards based on the BaSyx system architecture introduced in Section 2.3. Figure 4 depicts the deployment of the integrated solutions based on BaSyx. The strict design guidelines of BaSyx suggest

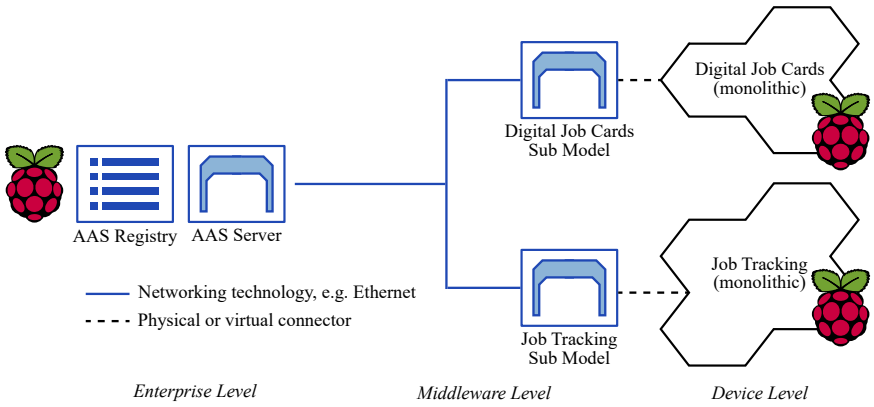


Figure 4. The combination of Job Tracking with Digital Job Cards based on the BaSyx system architecture. A single Raspberry Pi hosts both solutions, which are treated as legacy devices. Virtual connectors are used to abstract their submodels.

a *coordinator-based integration*, and since there are no specifications concerning data centralisation, the two databases remain separated. Compared to the considered reference architectures which require an explicit separation between resources and functionalities through modularising the low-cost digital system, *BaSyx does not require such modularisation since it allows for any asset to be encapsulated by an AAS, even a complete system.* Therefore, each low-cost digital system remains its monolithic design and is simply treated as a legacy application.

Facile. The integration based on BaSyx is more facile compared to an integration based on one of the considered reference architectures. Main interactions only take place via the middleware and all system-based communication remains within the system boundaries, which yields a particularly low coupling and high cohesion. Although the coordinators serve as a gateway for an integration based on reference architectures, messages can still reach the component level.

Successive. An integration based on BaSyx is less successive than an integration based on one of the considered reference architectures. While complete systems can be added without much development effort, their monolithic design makes adding further components more difficult.

Scalable. BaSyx is as scalable as the considered reference architectures when integrating systems. Their communication scheme is similar and enables the integration of numerous systems Schnicke et al. (2022a). If a component needs to be added to a system which is already connected to a shell, a new submodel can simply be added instead of modifying the existing interface, which may be less laborious.

Seamless. Since systems are not modularised, only complete systems can be integrated and disintegrated seamlessly when using BaSyx.

Reliable. The integration based on BaSyx is as reliable as an integration based Shoestring, WoT or PROSA/Erlang, since it relies on the same integration strategy.

Transparent. BaSyx can only guarantee transparency at a system level. Due to the monolithic design, there is no information available about the interactions of the components within a system.

In summary, while an integration based on the BaSyx system architecture achieves similar levels of scalability and reliability, reference architectures are more beneficial and should therefore be preferred when integrating the low-cost digital systems considered in this study.

6. Discussion

This section analyses the results of the conceptual design and evaluation of integration strategies, and the deployment and validation of the reference architecture driven integration. Specifically, we first perform a comparative analysis of the conceptual designs to outline practical implications when adopting one of the selected integration strategies. We then describe the key differences between an integration based on a reference architecture and system architecture, and describe integration scenarios in which system architectures should be preferred.

6.1. *Practical implications of adopting the integration strategies*

While this study has identified integration strategies that are most likely to be implemented at low cost, there may be cases which are not constrained by cost and where a different combination of strategies should be used. For example, for an integration of multiple digital systems with an existing centralised IT infrastructure, the integration effort may be less important than making the data and functions of these systems available to this central system. For this reason, we perform a comparative analysis of the conceptual designs to outline the practical implications developers need to consider when adopting one of the selected integration strategies (IS1-5) as per Figures 2(a) to 2(e) for Shoestring, WoT and PROSA/Erlang.

IS1: Data warehousing. Data warehousing requires additional capabilities for the architectural elements of the three reference architectures. Specifically, the data sources of the separate systems need to be physically integrated, a logical query schema is required and ETL pipelines need to be setup. Further, large data volumes, such as historical data, might require dedicated hardware. However, based on previous deployments and use cases (Swert et al. 2006; Hawkridge, Basson, and Kruger 2019; Hawkridge et al. 2021; Lagally et al. 2021), Shoestring, WoT and PROSA/Erlang typically develop systems in a distributed manner, where individual systems contain their data in their own storage, and thus, data centralisation may require much effort.

IS2: Virtual integration. Integrating data sources virtually requires a mediated schema and semantic mappings to the individual sources for Shoestring, WoT and PROSA/Erlang. Since these mandate a distributed system design, virtual integration is likely to require little development effort. Digital systems could be initially developed with distributed data sources with the option to shift to virtual integration. However, allowing both implies that only well structured data can be handled to reduce the effort of constructing the different views on the shared data for the mediated schema.

IS3: Coordinator-based integration. As shown above, coordinators for Shoestring can be developed by reusing service wrappers on a system level, that manage the data and functions by treating them as reactive resources. Designers can switch coordinators into an active or passive mode to allow for a successive integration and enable an evolution towards larger more complex environments. Passive coordinators require the designer to manually select data streams of interest from other service modules. Small systems benefit from this form of static configuration because it yields a fast development and predictable communication patterns. Active coordinators, on the other hand, control the application by managing data and resources autonomously. Shoestring allows for both active and passive coordinators to be used concurrently due to its loosely coupled provider-consumer communication. Conversely, WoT only exposes resources and does not mandate how consumers interact with them. Any form of coordination of resources needs to be performed by consumers. Hence, an additional coordinator element would yield a design overhead. Furthermore, coordinators for systems based on PROSA/Erlang are already represented by order holons. Besides managing the resources of their own system, order holons are capable of communicating with order holons of other systems.

IS4: Integration via a common service bus. There are several implications when extending the service bus. For Shoestring, more messages need to be handled by the same bus. In particular, the scalability is reduced due to fan-outs and broker bottlenecks as a result from asynchronous one-to-many or many-to-many communication, and request-response messaging suffers from increased latency because of the load caused by the additional system. Furthermore, a prioritisation scheme may be required to improve the system performance, which, for example, prioritises intra over inter-system messages or proposes a one-to-many publish-subscribe model, where the number of publisher nodes is limited. Conversely, WoT does not enforce system boundaries and enables consumers to interact with any resource available in the network. Therefore, an extra service bus or broker to interact with resources of other systems is not required if both are part of the same network. Similarly, the key architectural elements of PROSA/Erlang communicate through multiple Erlang runtime systems, so-called nodes, that exchange messages via TCP/IP sockets. Due to Erlang's distributed nature, nodes of a new system can be added without affecting the operation of other systems. An additional bus to interact with holons of other systems is not required. However, the order holons of each individual system need to negotiate how resources, data and functions are shared. Resource holons can take part in this negotiation as well since they are aware of their capability and capacity.

IS5: Integration via a central application layer. There are a number of implications when deploying a central application layer for Shoestring, WoT and PROSA/Erlang, which result from the three integration strategies used: primarily, the data warehouse requires a logical schema and ETL pipelines to be setup for each system. Further, the additional messages of the integrated systems lead to an overhead for the central controller, which may become a bottleneck for larger more complex systems. Finally, the service bus may require a custom prioritisation scheme to compensate for the lower system performance caused by this overhead.

6.2. *Comparison of a system architecture and reference architecture driven integration*

When using an architectural design approach to support the integration of digital systems, developers need to consider the different types of architectures and the integration scenario at hand. There are three main differences between an integration based on a system architecture compared to an integration based on one of the considered reference architectures for low-cost digital systems: first, for a system architecture driven integration, systems do not need to be modularised and packaged into architectural elements. The monolithically designed system is simply treated as a legacy device. Second, system architectures are likely to provide more specific design and implementation guidelines. In some cases, a software framework is given which facilitates the implementation of the structure, components and key functions of the architecture. Third, system architectures do not offer a structured template applicable to multiple types of systems. Designers have to create a custom integration design for each individual system, which reduces the reusability of components. While standardisation would increase the reusability, it would also make the system architecture more abstract. Based on these differences, we argue that there are *two integration scenarios where system architectures should be preferred*:

- **Integration with external elements.** External elements, such as legacy devices, cannot be easily modularised and packaged into architectural elements. In this case, reference architectures do provide any additional benefit over a system architecture, and developers should use system architectures as they are likely to require less integration effort due to their detailed design guidelines. For example, a simple legacy device consist of a machine, a control function to turn the machine on or off, and an embedded user interface showing the machine status. For BaSyx, developers would simply need to construct the interface to the control function and add this to a submodel within an AAS. In contrast, Shoestring and PROSA/Erlang would be incapable to wrap all features into architecture elements and would therefore not be able to model the legacy device accurately. For WoT, only the control functions can be abstracted to a Thing. However, the designer would need to further construct the type of interaction between this exposed legacy device and other consumers, whereas for BaSyx the shell manager handles the interaction autonomously upon initialisation.
- **The integrated system design is known beforehand.** If the designer knows in advance the number of systems that need to be connected, a system architecture may support a particularly effective integration of low-cost digital systems. While this integration scenario may occur in very specific situations, it is highly likely that systems will be required to evolve over time with the introduction of new technologies and production requirements. Reference architectures are more beneficial if further integration is desired but the number and types of additional systems and components is unknown. In this case, the template provided by a reference architecture enables a more structured development process.

Based on the exploration of trying to understand different integration approaches, we propose the following *reference architecture driven integration approach* for future consideration of integrating low-cost digital manufacturing systems: (1) select a reference architecture relevant to low-cost digital systems, and (2) perform the integration of low-cost digital systems by implementing *coordinators in combination with a common service bus and virtual data integration* for that reference architecture. Alterna-

tively, developers should rely on system architectures when it is necessary to integrate digital systems with external elements, or the integrated design is known beforehand.

7. Conclusions

Although there are many studies which separately investigate the role of reference architectures in design and challenges in system integration, none of these studies have examined the role of reference architectures in system integration. This paper set out to study the use of reference architectures as a means for supporting the integration of low-cost digital manufacturing systems. Based on common low-cost digital system design challenges, such as the need for a modular design with standardised interfaces and a successive development process (RQ1), we first defined success criteria for low-cost integration of digital systems. Then, we selected three detailed operational reference architectures relevant to low-cost digital manufacturing systems, namely Shoestring, WoT and PROSA/Erlang (RQ2). These reference architectures were chosen as exemplary design approaches to verify the results of this study. Next, we identified common integration strategies that have the potential to support low-cost digital manufacturing system integration (RQ3). Finally, we conceptually designed and evaluated each strategy by applying the success criteria to determine the the most effective integration strategy. The evaluation suggests that *coordinators in combination with a common service bus and virtual data integration* are likely to be most effective for integrating low-cost digital manufacturing systems (RQ4). For validation, a trial deployment for each reference architecture was developed for an industrially relevant application, and compared to an integration based on a system architecture. Based on the study of this paper, we have proposed the following steps as a *reference architecture driven integration approach* to effectively combine low-cost digital systems: (1) select a reference architecture relevant to low-cost digital systems, and (2) integrate these systems by implementing the recommended integration strategy for that reference architecture. Alternatively, developers should prefer system architectures when integrating systems with external elements, or the integrated design is known beforehand. The main contributions of this study include (a) an analysis of the capabilities of (reference and system) architectures to support a systematic integration of digital systems, and (b) an evaluation of the effectiveness of different integration strategies particularly with regard to low-cost digital systems.

The majority of integration approaches proposed are ad-hoc and require much development effort. This paper has shown that reference architectures provide sufficient structure and support an effective integration of low-cost digital systems. In particular, the two main challenges of integration - distributed heterogeneous data sources and the lack of a common interface - can be met by using operational detailed reference architectures in combination with an appropriate integration strategy. While we focused on the most elementary form of integration, the proposed reference architecture driven integration is also capable of forming more complex digital system combinations, and thus it can form part of a broader integration, such as an entire factory supported by low-cost digital solutions.

This research work is subject to a number of limitations: first, we only focus on integrating systems derived from the same reference architecture and do not study the integration with legacy systems or systems designed based on different reference architectures. Second, this study is concerned with particularly low-cost digital manufacturing systems. For integrating digital systems that are not constrained by cost, a

different integration approach may be required. Third, this study has only performed a qualitative evaluation of the integration effort. A quantitative evaluation of the integration effort (e.g. number of required connections between components) for each reference architecture would provide additional insights, such as the strengths and shortcomings in the design process of a reference architecture. Fourth, the effectiveness of the recommended strategy has only been demonstrated for a single industrial use case. More insights could be gained from analysing and comparing additional use cases, where different types of data and functions of low-cost digital systems need to be shared, and the systems are subject to different performance requirements.

There are three main pathways that arise from the research conducted here: first, there is a need for investigating how reusable the proposed reference architecture driven approach is under different circumstances, such as the integration of systems based on reference architectures with external elements. Second, there is need for a systematic pathway to integration, which supports SMEs in developing and deploying multiple low-cost digital systems. Finally, a knowledge base could be developed to visualise potential data flows between systems and resources that could be shared among them. Such a directed graph would facilitate the integration process by recommending systems that can be easily integrated with.

Disclosure statement

There is no conflict of interest to declare.

Funding

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC: EP/R032777/1).

References

Beregi, Richárd, Gianfranco Pedone, Borbála Háy, and József Váncza. 2021. "Manufacturing Execution System Integration through the Standardization of a Common Service Model for Cyber-Physical Production Systems." *Applied Sciences* 11 (16). `https://www.mdpi.com/2076-3417/11/16/7581`.

Chen, Shang Liang, Yun Yao Chen, and Chiang Hsu. 2014. "A new approach to integrate internet-of-things and software-as-a-service model for logistic systems: A case study." *Sensors (Switzerland)* 14: 6144–6164.

Collina, Matteo, Giovanni Emanuele Corazza, and Alessandro Vanelli-Coralli. 2012. "Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST." In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 36–41.

Coronado, Pedro Daniel Urbina, Roby Lynn, Wafa Louhichi, Mahmoud Parto, Ethan Wescoat, and Thomas Kurfess. 2018. "Part data integration in the Shop Floor Digital Twin: Mobile and cloud technologies to enable a manufacturing execution system." *Journal of Manufacturing Systems* 48: 25–33.

Covanich, Wutthiphat, Duncan McFarlane, James Brusey, and Amro M Farid. 2007. "Integrating a New Machine into an Existing Manufacturing System." In *5th IEEE International Conference on Industrial Informatics*, 861–866.

Cruz-Correia, Ricardo João. 2010. "Implementation, monitoring and utilization of an inte-

grated hospital information system - lessons from a case study." In *Studies in Health Technology and Informatics*, Vol. 160, 238–241.

Doan, AnHai, Alon Halevy, and Zachary Ives. 2012. *Principles of Data Integration*. 1st ed. Morgan Kaufmann.

Erbe, Heinz H. 2002. "Low cost intelligent automation in manufacturing." In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, Vol. 15, 373–378.

Fang, Shifeng, Lida Xu, Yunqiang Zhu, Yongqiang Liu, Zhihui Liu, Huan Pei, Jianwu Yan, and Huifang Zhang. 2015. "An integrated information system for snowmelt flood early-warning based on internet of things." *Information Systems Frontiers* 17: 321–335.

Feng, Shaw C., and Yuhwei Yang. 1995. "A dimension and tolerance data model for concurrent design and systems integration." *Journal of Manufacturing Systems* 14 (6): 406–426.

Filip, Florin Gheorghe. 2011. "Designing and Building Modern Information Systems; A Series of Decisions to Be Made." *Computer Science Journal of Moldova* 19: 2011.

Gudivada, Venkat N., and Jagadeesh Nandigam. 2005. "Enterprise application integration using extensible web services." In *IEEE International Conference on Web Services (ICWS)*, Vol. 2005, 41–48.

Hasani, S., A. Sadeghi-Niaraki, and M. Jelokhani-Niaraki. 2015. "Spatial data integration using ontology-based approach." In *International Conference on Sensors Models in Remote Sensing Photogrammetry*, Vol. 40, 293–296.

Hasselbring, Wilhelm. 2000. "Information system integration." *Communications of the ACM* 43: 32–38.

Hawkridge, G. T., A. H. Basson, and K. Kruger. 2019. "Comparison of Erlang/OTP and JADE implementations for standby redundancy in a holonic controller." *International Journal of Computer Integrated Manufacturing* 32: 1207–1230.

Hawkridge, Gregory, Marco Perez Hernandez, Lavindra de Silva, German Terrazas, Yedige Tlegenov, Duncan McFarlane, and Alan Thorne. 2019. "Tying Together Solutions for Digital Manufacturing: Assessment of Connectivity Technologies Approaches." In *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, .

Hawkridge, Gregory, Duncan McFarlane, Jan Kaiser, Lavindra de Silva, and German Terrazas. 2022. "Designing Shoestring Solutions: An Approach for Designing Low-Cost Digital Solutions for Manufacturing." In *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future*, 249–262.

Hawkridge, Gregory, Anandarup Mukherjee, Duncan McFarlane, Yedige Tlegenov, Ajith K. Parlikad, Nicholas J. Reyner, and Alan Thorne. 2021. "Monitoring on a shoestring: Low cost solutions for digital manufacturing." *Annual Reviews in Control* 51: 374–391.

Hepner, M., R. Gamble, M. Kelkar, L. Davis, and D. Flagg. 2006. "Patterns of conflict among software components." *Journal of Systems and Software* 79: 537–551.

Hilario, Manuel, Doris Esenarro, Hugo Vega, and Ciro Rodriguez. 2021. "Integration Of The Enterprise Information To Facilitate Decision Making." *Journal of Contemporary Issues in Business and Government* 27.

Hoxha, Valmir, Ines Bula, Muzafer Shala, and Edmond Hajrizi. 2016. "Cost-Oriented Open Source Automation Potential Application in Industrial Control Applications." In *IFAC-PapersOnLine*, Vol. 49, 212–214.

IEC 62264-1. 2003. "Enterprise-control system integration - Part 1: Models and terminology." .

IEC TR 62541-1. 2020. "OPC unified architecture - Part 1: Overview and concepts." .

IEC/PAS 63088. 2017. "Smart manufacturing - Reference architecture model industry 4.0 (RAMI4.0)." .

Jones, Albert T., and Charles R. McLean. 1986. "A proposed hierarchical control model for automated manufacturing systems." *Journal of Manufacturing Systems* 5 (1): 15–25.

Kaiser, Jan, Gregory Hawkridge, Gokcen Yilmaz, and Duncan McFarlane. 2022a. "A Low-cost Integration Approach for Distributed Manufacturing Information Systems." In *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Vol. 5, 763–769.

Kaiser, Jan, Zhengyang Ling, Gokcen Yilmaz, Duncan McFarlane, and Gregory Hawkridge. 2022b. "Configurable Solutions for Low-Cost Digital Manufacturing: a Building Block Approach." In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–9.

Kaiser, Jan, Duncan McFarlane, and Gregory Hawkridge. 2022. "Review and Classification of Digital Manufacturing Reference Architectures." In *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future*, 231–247.

Kaiser, Jan, Duncan McFarlane, Gregory Hawkridge, Pascal André, and Paulo Leitão. 2023. "A review of reference architectures for digital manufacturing: Classification, applicability and open issues." *Computers in Industry* 149: 103923.

Kaiser, Jan, German Terrazas, Duncan McFarlane, and Lavindra de Silva. 2021. "Towards low-cost machine learning solutions for manufacturing SMEs." *AI and Society* .

Kruger, Karel, and Anton Basson. 2017. "Erlang-based control implementation for a holonic manufacturing cell." *International Journal of Computer Integrated Manufacturing* 30: 641–652.

Kuhn, Thomas, Pablo Oliveira Antonino, and Frank Schnicke. 2020. "Industrie 4.0 Virtual Automation Bus Architecture." In *Software Architecture*, 477–489.

Lagally, Michael, Ryuichi Matsukura, Toru Kawaguchi, Kunihiko Toumura, and Kazuo Kajimoto. 2021. "Web of Things (WoT) Architecture 1.1 - W3C Editor's Draft 27 May 2021." `https://w3c.github.io/wot-architecture/`.

Land, Rikard, and Ivica Crnkovic. 2003. "Software Systems Integration and Architectural Analysis - A Case Study." In *IEEE International Conference on Software Maintenance, ICSM*, 338–347.

Macias-Aguayo, Jaime, Duncan McFarlane, Benjamin Schönfuß, and Liz Salter. 2022. "A catalogue of digital solution areas for logistics SMEs." *IFAC-PapersOnLine* 55: 1828–1833.

Mahmoodpour, Mehdi, and Andrei Lobov. 2019. "A knowledge-based approach to the IoT-driven data integration of enterprises." In *9th Conference on Learning Factories*, Vol. 31, 283–289.

McFarlane, Duncan, Svetan Ratchev, Alan Thorne, Ajith Kumar Parlikad, Lavindra de Silva, Benjamin Schönfuß, Greg Hawkridge, German Terrazas, and Yedige Tlegenov. 2020. "Digital Manufacturing on a Shoestring: Low Cost Digital Solutions for SMEs." In *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future*, 40–51.

Modoni, G. E., M. Doukas, W. Terkaj, M. Sacco, and D. Mourtzis. 2017. "Enhancing factory data integration through the development of an ontology: from the reference models reuse to the semantic conversion of the legacy models." *International Journal of Computer Integrated Manufacturing* 30 (10): 1043–1059. `https://doi.org/10.1080/0951192X.2016.1268720`.

Modoni, G. E., A. Trombetta, M. Veniero, M. Sacco, and D. Mourtzis. 2019. "An event-driven integrative framework enabling information notification among manufacturing resources." *International Journal of Computer Integrated Manufacturing* 32 (3): 241–252. `https://doi.org/10.1080/0951192X.2019.1571232`.

Molina, A., C. A. Rodriguez, H. Ahuett, J. A. Cortés, M. Ramírez, G. Jiménez, and S. Martinez. 2005. "Next-generation manufacturing systems: key research issues in developing and integrating reconfigurable and intelligent machines." *International Journal of Computer Integrated Manufacturing* 18 (7): 525–536.

Mostefai, Sihem, Abdelaziz Bouras, and Mohamed Batouche. 2005. "Data Integration in a PLM Perspective for Mechanical Products." *The International Arab Journal of Information Technology* 2.

Mularz, Diane E. 1995. "Pattern-Based Integration Architectures." *Pattern Languages of Program Design* 441–452.

Müller, Timo, Simon Kamm, Andreas Löcklin, Dustin White, Marius Mellinger, Nasser Jazdi, and Michael Weyrich. 2022. "Architecture and knowledge modelling for self-organized reconfiguration management of cyber-physical production systems." *International Journal of Computer Integrated Manufacturing* 0 (0): 1–22. `https://doi.org/10.1080/0951192X.2022.2121425`.

Nilsson, Erik G., Else K. Nordhagen, and Gro Oftedal. 1990. “Aspects of systems integration.” In *Proceedings of the First International Conference on Systems Integration*, 434–443.

Patel, Jayesh. 2019. “Overcoming data silos through big data integration.” *International Journal of Information Technology and Management (IJIT)* 4.

Rojko, Andreja. 2017. “Industry 4.0 concept: Background and overview.” *International Journal of Interactive Mobile Technologies* 11: 77–90.

Roşca, Doina, Logica Bănică, and Mirela Sîrbu. 2010. “Building Successful Information Systems-a Key for Successful Organization.” *Economics and Applied Informatics* .

Sabooniha, Nazanin, Danny Toohey, and Kevin Lee. 2012. “An Evaluation of Hospital Information Systems Integration Approaches.” In *International Conference on Advances in Computing, Communications and Informatics (ICACCI’12)*, 1307.

Sahara, Chelinka Rafiesta, and Ammar Mohamed Aamer. 2021. “Real-time data integration of an internet-of-things-based smart warehouse: a case study.” *International Journal of Pervasive Computing and Communications* .

Schnicke, Frank, Ashfaqul Haque, Thomas Kuhn, Daniel Espen, and Pablo Oliveira Antonino. 2022a. “Architecture Blueprints to Enable Scalable Vertical Integration of Assets with Digital Twins.” In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8.

Schnicke, Frank, Thomas Kuhn, Tobias Klausmann, Sten Grüner, and Daniel Porta. 2022b. “Architecture Blueprints for the Application of the Industry 4.0 Asset Administration Shell.” In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8.

Schönfuß, Benjamin, Duncan McFarlane, Gregory Hawkridge, Liz Salter, Nicky Athanassopoulou, and Lavindra de Silva. 2021. “A catalogue of digital solution areas for prioritising the needs of manufacturing SMEs.” *Computers in Industry* 133.

Swert, Kris De, Paul Valckenaers, Bart Saint German, Paul Verstraete, Hadeli, and Hendrik Van Brussel. 2006. “Coordination and control for railroad networks inspired by manufacturing control.” In *Proceedings - DIS 2006: IEEE Workshop on Distributed Intelligent Systems - Collective Intelligence and Its Applications*, Vol. 2006, 201–206.

Van Brussel, Hendrik, Jo Wyns, Paul Valckenaers, Luc Bongaerts, and Patrick Peeters. 1998. “Reference architecture for holonic manufacturing systems: PROSA.” *Computers in Industry* 37: 255–274.

Wilamowska, Katarzyna, Thai Le, George Demiris, and Hilaire Thompson. 2013. “Using commercially available tools for multifaceted health assessment: Data integration lessons learned.” *CIN - Computers Informatics Nursing* 31: 329–334.

Yang, Hui-Mei, and FangLieh Victor Lu. 2005. “Integrating inter- and extra-enterprise applications using web services.” *Review of Business* 26: 3–9.

Zhang, Yingfeng, Geng Zhang, Junqiang Wang, Shudong Sun, Shubin Si, and Teng Yang. 2015. “Real-time information capturing and integration framework of the internet of manufacturing things.” *International Journal of Computer Integrated Manufacturing* 28 (8): 811–822.