MDPI

Article

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

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: This documents provides supplementary data to the reference architecture evaluation paper. Specifically, it describes the design verification and validation of the Job Tracking (baseline) solution for Shoestring, WoT and PROSA/Erlang, and includes all 18 design structure matrices for the selected low-cost digital design scenarios.

1. Design *veri ication* of the Job Tracking solution

Table S1. Design verification of the Job Tracking solution based on Shoestring. A checklist is used to assess the architectural compliance.

Architecture descriptions and implementation guidelines of Shoestring [1,2]	✓or × or N/A
Each solution satisfies a single solution area in the Shoestring catalogue	✓
Each solution is made up of service modules	✓
Each service module is categorised as only one of the six service module types	✓
Service modules communicate with one another using a shoestring approved service layer	✓
Service modules communicate using a producer-consumer model	✓
Producers make data streams/sources available	✓
Consumers fetch/subscribe to that data	✓
Producers require no knowledge of the destination of their produced data, or which consumers are attached	✓
Service modules should support synchronous request-response communication	N/A
For static data	N/A
For control requests	N/A
Service modules should support asynchronous (1-to-n) multicast (or pub-sub) communication	✓
For regularly changing data	✓
For events/error logs	✓
Service modules are made up of building blocks	✓
Building blocks can communicate directly with other building blocks in the same service module	✓
Building blocks may not communicate directly with other building blocks in different service modules	✓
Communication between building blocks in different service modules must happen over the service layer (via the service wrapper)	✓
Each building block must be characterised by one of the building block types in the shoestring building block type catalogue	✓

Table S2. Design verification of the Job Tracking solution based on WoT. A checklist is used to assess the architectural compliance.

Architecture descriptions and implementation guidelines of WoT [3]	✓or × or N/A
The WoT architecture should enable mutual interworking of different eco-systems using web technology	✓
The WoT architecture should be based on the web architecture using RESTful APIs (e.g. HTML, PHP)	✓

The WoT architecture should allow to use multiple payload formats which are commonly used in the web (e.g. JSON, XML)	✓
The WoT architecture must enable different device architectures and must not force a client or server implementation of system components	✓
Flexibility: the WoT abstract architecture should be able to be mapped to and cover all variations of physical device configurations	N/A
Compatibility: WoT should provide a bridge between these existing and developing IoT solutions and Web technology based on WoT concepts	N/A
Scalability: WoT must be able to scale for IoT solutions that incorporate thousands to millions of devices	✓
Interoperability: WoT must provide interoperability across device and cloud manufacturers	✓
Capable of reading thing's status information	✓
Capable of updating thing's status information which might cause actuation	✓
Capable of subscribing to, receiving and unsubscribing to notifications of changes of the thing's status information	N/A
Capable of invoking functions with input and output parameters which would cause certain actuation or calculation	N/A
Capable of subscribing to, receiving and unsubscribing to event notifications that are more general than just reports of state transitions	N/A
The WoT architecture should allow clients to know thing's attributes, functionalities and their access points, prior to access to the thing itself	✓
The WoT architecture should allow clients to search things by its attributes and functionalities	N/A
The WoT architecture should allow semantic search of things providing required functionalities based on a unified vocabulary, regardless of naming of the functionalities	N/A
Descriptions should be not only human-readable, but also machine-readable	✓
Descriptions should allow semantic annotation of its structure and described contents	✓
Description should be able to be exchanged using multiple formats which are commonly used in the web	N/A
The WoT architecture should allow describing thing's attributes such as name, explanation, version of spec, format and description itself, links to other related things and metadata information	✓
Such descriptions should support internationalisation	✓
WoT architecture should allow describing thing's functionalities which are shown above	✓
The WoT architecture should support multiple web protocols which are commonly used	✓
Such protocols include protocols commonly used in the internet and protocols commonly used in the local area network	✓
The WoT architecture should allow using multiple web protocols to access to the same functionality	✓
The WoT architecture should allow using a combination of multiple protocols to the functionalities of the same thing (e.g., HTTP and WebSocket)	N/A
The WoT architecture should support a wide variety of thing capabilities such as edge devices with resource restrictions and virtual things on the cloud, based on the same model	✓
The WoT architecture should support multiple levels of thing hierarchy with intermediate entities such as gateways and proxies	N/A
The WoT architecture should support accessing things in the local network from the outside of the local network (the internet or another local network), considering network address translation	N/A
The WoT architecture should allow describing applications for a wide variety of things such as edge device, gateway, cloud and UI/UX device, using web standard technology based on the same model	N/A
The WoT architecture should allow mapping of legacy IP and non-IP protocols to web protocols, supporting various topologies, where such legacy protocols are terminated and translated	N/A
The WoT architecture should allow transparent use of existing IP protocols without translation, which follow RESTful architecture	N/A
The WoT architecture must not enforce client or server roles on devices and services (IoT devices can be either clients or servers, or both, depending on the system architecture)	✓
Access to devices is made using a description of their functions and interfaces	✓
Applications need to be able to generate and use network and program interfaces based on metadata (descriptions)	✓
A twin has to produce a description for the virtual device and make it externally available	N/A

Directories can provide functionalities to allow devices and twins themselves automatically or the users to manually register the descriptions, which creates devices searchable by external entities	N/A
Security information related to devices and virtual devices needs to be described in device descriptions	N/A
The Thing description metadata MUST be a WoT Thing Description (TD)	✓
Consumers MUST be able to parse and process the TD representation format, which is based on JSON	✓
To be a Thing, however, at least one TD representation MUST be available	✓
WoT Thing Descriptions MAY link to other Things and other resources on the Web to form a Web of Things	N/A
An identifier in the WoT Thing Description MUST allow for the correlation of multiple TDs representing the same original Thing or ultimately unique physical entity	N/A
Things MAY be bundled together with a Consumer to enable Thing-to-Thing interaction	✓
The configuration of the Consumer behavior MAY be exposed through the Thing	✓
Things MAY offer three other types of Interaction Affordances defined by this specification: Properties, Actions, and Events	✓
The state exposed by a Property MUST be retrievable (readable), MAY be updated (writeable), and MAY choose to make Properties observable by pushing the new state after a change	✓
Properties MAY contain one data schema for the exposed state	✓
An Action MAY manipulate state that is not directly exposed, manipulate multiple Properties at a time, or manipulate Properties based on internal logic	✓
Invoking an Action MAY also trigger a process on the Thing that manipulates state (including physical state through actuators) over time	N/A
Actions MAY contain data schemas for optional input parameters and output results	✓
Events MAY be triggered through conditions that are not exposed as Properties	N/A
Events MAY contain data schemas for the event data and possible subscription control messages	N/A
For links, extension relation types MUST be compared as strings using a case-insensitive comparison	✓
For links, all-lowercase URIs SHOULD be used for extension relation types	N/A
Form contexts and submission targets MUST both be Internationalized Resource Identifiers (IRIs)	N/A
Form context and submission target MAY point to the same resource or different resources, where the submission target resource implements the operation for the context	N/A
For forms, well-known operation types MUST follow the ABNF and MUST be compared using a case-insensitive comparison	N/A
For forms, the set of predefined operation types MAY be augmented by Extension operation types chosen by applications	N/A
For forms, extension operation types MUST be URIs	N/A
For forms, extension operation types MUST be compared as strings using a case-insensitive comparison	N/A
For forms, all-lowercase URIs SHOULD be used for extension operation types	N/A
For forms, the request method MUST identify one method of the standard set of the protocol identified by the submission target URI scheme	N/A
Form fields are optional and MAY further specify the expected request message for the given operation	N/A
Form fields MAY depend on the protocol used for the submission target as specified in the URI scheme	N/A
Interaction Affordances MUST include one or more Protocol Bindings	✓
Protocol Bindings MUST be serialized as hypermedia controls to be self-descriptive on how to activate the Interaction Affordance	✓
The hypermedia controls MUST originate from the authority managing the Thing that is providing the corresponding Interaction Affordance	✓
The hypermedia controls MAY be cached outside the Thing and used for offline processing if caching metadata is available to determine the freshness	N/A
Eligible protocols for W3C WoT MUST have an associated URI scheme that is registered with IANA	✓
Eligible protocols for W3C WoT MUST be based on a standard set of methods that are known a priori	✓
All data (a.k.a. content) exchanged when activating Interaction Affordances MUST be identified by a media type in the Protocol Binding, e.g. application/json	✓
Protocol Bindings MAY have additional information that specifies representation formats in more detail than the media type alone	N/A
Corresponding Interaction Affordances SHOULD declare a data schema to provide more detailed syntactic metadata for the data exchanged	✓

Servient software stack (representation of a Thing called Exposed Thing including its WoT Interface available to Consumers of the Thing) are used to enable direct/indirect communication between Things and consumers	✓
TD generated by an Intermediary MAY contain interfaces for other communication protocols	N/A

Table S3. Design verification of the Job Tracking solution based on PROSA/Erlang. A checklist is used to assess the architectural compliance.

Architecture descriptions and implementation guidelines of PROSA/Erlang [4,5]	✓or × or N/A
Holon structure: an autonomous and co-operative building block of a manufacturing system consisting of an information processing part and often a physical processing part	✓
Autonomy: entities are capable of creating and controlling the execution of its own plans and strategies	✓
Cooperation: a set of entities develops mutually acceptable plans and executes these plans	✓
Holarchy: a system of holons that can cooperate to achieve a goal or objective by defining basic rules for cooperation of the holons and thereby limiting their autonomy	✓
Aggregation: a clustered set of related holons that forms a bigger holon with its own identity	N/A
Specialisation: separate basic holons into product holons (hold process and product knowledge), resource holons (offer production capacity and functionality), and order holons (manages service provided by resource holons via exchanging process execution information)	✓
Staff holons: assist basic holons in performing their task	N/A
Inter-holon and intra-holon communication is implemented in Erlang	✓
Each RH consists of four elements: a communication component, an agenda manager, an execution component, and an interface component	✓
The agenda manager can be implemented through a generic server or a finite state machine	✓
The execution component is implement via a finite state machine	✓
The interface component can be implemented by using OTP functions for TCP/IP or UDP communication) or through ports (or linked-in port drivers)	✓
The intra-holon components of OHs are adapted from the RH structure	✓
Data flows through OH (RH to RH communication is not specified)	✓

2. Design *validation* of the Job Tracking solution

Table S4. Functionality testing as part of the design validation of the Job Tracking solution for Shoestring, WoT and PROSA/Erlang. The functionality is tested by checking if all specifications of the Job Tracking solution have been met. These specifications have been identified through conducting an in-company workshop.

Services	**Functionality**	**Shoestring**	**WoT**	**PROSA/Erlang**
Sensing	Barcode scanning	1	1	1
Data	States of jobs	1	1	1
	Planned times	0	0	0
	Where a job is at a specific time	1	1	1
Commands	Login	0	0	0
Information	Actual times vs planned times	0	0	0
Action	Scan barcode	1	1	1
	Search jobs by date	1	1	1
	Search jobs by job number	1	1	1
	Search jobs by machine number	0	0	0
	Open user specific dashboards	1	1	1
		63.6%	63.6%	63.6%

Table S5. Interoperability testing of the Job Tracking solution designed based on Shoestring. The interoperability is tested for the main architectural elements of the design, namely the data collection (DC), data storage (DS) and user interface service modules (UI1, UI2, UI3). Its key concerns revolve around the exchange and interpretability of data among service modules and complete digital systems.

	Checks	Source	Target	1 or 0	Score
Intra-system interoperability	Can the source exchange information with the target?	DC	DS	1	1
			UI1	0	0
			UI2	0	0
			UI3	0	0
		DS	DC	1	1
			UI1	1	1
			UI2	1	1
			UI3	1	1
		UI1	DC	0	0
			DS	1	1
			UI2	0	0
			UI3	0	0
		UI2	DC	0	0
			DS	1	1
			UI1	0	0
			UI3	0	0
		UI3	DC	0	0
			DS	1	1
			UI1	0	0
			UI2	0	0
	Are special/custom *hardware* drivers needed for operating components?	DC		0	1
		DS		0	1
		UI1		0	1
		UI2		0	1
		UI3		0	1
	Are special/custom *software* drivers needed for operating components?	DC		0	1
		DS		0	1
		UI1		0	1
		UI2		0	1
		UI3		0	1
	Can information be interpreted between services directly (without drivers)?			1	1
	Are data exchange formats between services common?			1	1
	Are parameter/variable names common across services?			0	0
Inter-system interoperability	Can the system integrate with the target system directly (without converters/drivers)?	N/A	N/A	0	0
	Can shared resources be accessed by the system?			0	0
	Can shared functions be accessed by the system?			0	0
	Any IP-protected technologies involved?			0	1
	Any licensed technologies involved?			0	1
					57.9%

Table S6. Interoperability testing of the Job Tracking solution designed based on WoT. The interoper-ability is tested for the main architectural elements of the design, namely the barcode scanner thing (BST), the database thing (DBT), the user interface things (UIT1, UIT2, UIT3), and the consumer (C). Its key concerns revolve around the exchange and interpretability of data among things and complete digital systems.

	Checks	Source	Target	1 or 0	Score
	Can the source exchange information with the target?	C	BST	1	1
			DBT	1	1
			UIT1	0	0
			UIT2	0	0
			UIT3	0	0

		BST	C	1	1
			DBT	0	0
			UIT1	0	0
			UIT2	0	0
			UIT3	0	0
		DBT	C	1	1
			BST	0	0
			UIT1	1	1
			UIT2	1	1
			UIT3	1	1
		UIT1	C	0	0
			BST	0	0
			DBT	1	1
			UIT2	0	0
			UIT3	0	0
		UIT2	C	0	0
			BST	0	0
			DBT	1	1
			UIT1	0	0
			UIT3	0	0
		UIT3	C	0	0
			BST	0	0
			DBT	1	1
			UIT1	0	0
			UIT2	0	0
Intra-system interoperability	Are special/custom *hardware* drivers needed for operating components?	C		0	1
		BST		0	1
		DBT		0	1
		UIT1		0	1
		UIT2		0	1
		UIT3		0	1
	Are special/custom *software* drivers needed for operating components?	C		0	1
		BST		0	1
		DBT		0	1
		UIT1		0	1
		UIT2		0	1
		UIT3		0	1
	Can information be interpreted between services directly (without drivers)?			1	1
	Are data exchange formats between services common?			1	1
	Are parameter/variable names common across services?			1	1
Inter-system interoperability	Can the system integrate with the target system directly (without converters/drivers)?	N/A	N/A	0	0
	Can shared resources be accessed by the system?			0	0
	Can shared functions be accessed by the system?			0	0
	Any IP-protected technologies involved?			0	1
	Any licensed technologies involved?			0	1
					54.0%

Table S7. Interoperability testing of the Job Tracking solution designed based on PROSA/ Erlang. The interoperability is tested for the main architectural elements of the design, namely the barcode scanner (BSRH), database (DBRH), and user interface resource holons (UIRH1, UIRH2, UIRH3), and the order holon (OH). Its key concerns revolve around the exchange and interpretability of data among holons and complete digital systems.

	Checks	**Source**	**Target**	**1 or 0**	**Score**
	Can the source exchange information with the target?	OH	BSRH	1	1
			DBRH	1	1
			UIRH1	1	1
			UIRH2	1	1

			UIRH3	1	1
		BSRH	OH	1	1
			DBRH	0	0
			UIRH1	0	0
			UIRH2	0	0
			UIRH3	0	0
		DBRH	OH	1	1
			BSRH	0	0
			UIRH1	0	0
			UIRH2	0	0
			UIRH3	0	0
		UIRH1	OH	1	1
			BSRH	0	0
			DBRH	0	0
			UIRH2	0	0
			UIRH3	0	0
		UIRH2	OH	1	1
			BSRH	0	0
			DBRH	0	0
			UIRH1	0	0
			UIRH3	0	0
		UIRH3	OH	1	1
			BSRH	0	0
			DBRH	0	0
			UIRH1	0	0
			UIRH2	0	0
Intra-system interoperability	Are special/custom *hardware* drivers needed for operating components?	OH		0	1
		BSRH		0	1
		DBRH		0	1
		UIRH1		0	1
		UIRH2		0	1
		UIRH3		0	1
	Are special/custom *software* drivers needed for operating components?	OH		0	1
		BSRH		0	1
		DBRH		0	1
		UIRH1		0	1
		UIRH2		0	1
		UIRH3		0	1
	Can information be interpreted between services directly (without drivers)?			1	1
	Are data exchange formats between services common?			1	1
	Are parameter/variable names common across services?			1	1
Inter-system interoperability	Can the system integrate with the target system directly (without converters/drivers)?	N/A	N/A	0	0
	Can shared resources be accessed by the system?			0	0
	Can shared functions be accessed by the system?			0	0
	Any IP-protected technologies involved?			0	1
	Any licensed technologies involved?			0	1
					54.0%

Table S8. Performance testing of the Job Tracking solution for Shoestring, WoT and PROSA/ Erlang. The performance is tested by selecting certain parameters based on the specifications of the solution identified through conducting an in-company workshop, and testing it against these parameters. The tests and results for all three reference architecture designs are the same.

Function	**Parameter**	**Range/value**	**Check**
Barcode scanning	Speed	Time spent scanning should not exceed 3 min per hour	The barcode appears on the dashboard in <5 sec, which allows up to 36 scans per operator per hour
Data storage	Update interval	Real-time	Real-time
User interface	Refresh rate	Real-time	Real-time

3. Shoestring design structure matrices

Design Structure Matrix Shoestring Baseline	Vibration Sensor	Sensor Service Module	Dashboard Service Module	Dashboard Server	Dashboard
Vibration Sensor	x				
Sensor Service Module	1	x			
Dashboard Service Module		1	x		
Dashboard Server			1	x	
Dashboard				1	x

Design Structure Matrix Shoestring Add feature	Vibration Sensor	Sensor Service Module	Analysis Service Module	Threshold	Dashboard Service Module	Dashboard Server	Dashboard
Vibration Sensor	x						
Sensor Service Module	1	x					
Analysis Service Module		1	x	1			
Threshold			1	x			
Dashboard Service Module		1	1		x		
Dashboard Server					1	x	
Dashboard						1	x

Design Structure Matrix Shoestring Add component	Vibration Sensor	Noise Filter	Sensor Service Module	Dashboard Service Module	Dashboard Server	Dashboard & Graph
Vibration Sensor	x					
Noise Filter		x	1			
Sensor Service Module	1	1	x			
Dashboard Service Module			1	x		
Dashboard Server				1	x	
Dashboard & Graph					1	x

Design Structure Matrix Shoestring Replace phys. resource	Temperature Sensor	Sensor Service Module	Dashboard Service Module	Dashboard Server	Dashboard
Temperature Sensor	x				
Sensor Service Module	1	x			
Dashboard Service Module		1	x		
Dashboard Server			1	x	
Dashboard				1	x

Design Structure Matrix Shoestring Replace virt. resource	Vibration Sensor	Sensor Service Module	GUI Service Module	GUI
Vibration Sensor	x			
Sensor Service Module	1	x		
GUI Service Module		1	x	
GUI			1	x

Design Structure Matrix Shoestring Integration	Barcode Scanner	Scanner Service Module	Database#1 Service Module	Database#1	UI#1 Service Module	UI#1	UI#2 Service Module	UI#2	UI#3 Service Module	UI#3	*Job Tracking Coordinator*	*Job Cards Coordinator*	Database#2 Service Module	Database#2	Dashboard Service Module	Dashboard
Barcode Scanner	x															
Scanner Service Module	1	x														
Database#1 Service Module		1	x													
Database#1			1	x												
UI#1 Service Module				1	x											
UI#1					1	x										
UI#2 Service Module				1			x									
UI#2							1	x								
UI#3 Service Module				1					x							
UI#3									1	x						
Job Tracking Coordinator		1									x					
Job Cards Coordinator											1	x				
Database#2 Service Module												1	x			
Database#2													1	x	1	
Dashboard Service Module												1		1	x	1
Dashboard															1	x

4. WoT design structure matrices

Design Structure Matrix WoT Baseline	Thing Description Server	Client	Vibration Sensor	Dashboard Server	Dashboard
Thing Description Server	x	1	1		
Client	1	x			
Vibration Sensor			x		
Dashboard Server	1			x	
Dashboard				1	x

Design Structure Matrix WoT Add feature	Thing Description Server	Client	Vibration Sensor & Threshold	Dashboard Server	Dashboard
Thing Description Server	x	1	1		
Client	1	x			
Vibration Sensor & Threshold			x		
Dashboard Server	1			x	
Dashboard				1	x

Design Structure Matrix WoT Add component	Thing Description Server	Client	Vibration Sensor	Noise Filter	Dashboard Server	Dashboard & Graph
Thing Description Server	x	1	1	1		
Client	1	x				
Vibration Sensor			x			
Noise Filter	1			x		
Dashboard Server	1				x	
Dashboard & Graph					1	x

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

Design Structure Matrix **WoT** **Replace virt. resource**	Thing Description Server	Client	Vibration Sensor	GUI
Thing Description Server	x	1	1	
Client	1	x		
Vibration Sensor			x	
GUI	1			x

Design Structure Matrix **WoT** **Integration**	Thing Description Server#1	Barcode Scanner	Database#1	UI#1	UI#2	UI#3	Websocket#1	Websocket#2	*Coordinator Client*	Thing Description Server#2	*Virtual* Scanner	Database#2	Dashboard	Websocket#3	Websocket#4	Websocket#5
Thing Description Server#1	x	1							1							
Barcode Scanner	1	x														
Database#1			x				1	1								
UI#1	1			x												
UI#2	1				x											
UI#3	1					x										
Websocket#1	1						x									
Websocket#2	1		1					x								
Coordinator Client	1								x							
Thing Description Server#2									1	x	1				1	
Virtual Scanner	1										x					
Database#2												x		1	1	1
Dashboard										1			x			
Websocket#3										1				x		
Websocket#4										1		1			x	
Websocket#5										1						x

5. PROSA/Erlang design structure matrices

Design Structure Matrix PROSA/Erlang Baseline	Start Node	Order Holon	Sensor Resource Holon	Vibration Sensor	Dashboard Resource Holon	Dashboard Server	Dashboard
Start Node	x						
Order Holon	1	x	1		1		
Sensor Resource Holon	1	1	x	1			
Vibration Sensor				x			
Dashboard Resource Holon	1	1			x		
Dashboard Server					1	x	
Dashboard						1	x

Design Structure Matrix PROSA/Erlang Add feature	Start Node	Order Holon	Sensor Resource Holon	Vibration Sensor & Threshold	Dashboard Resource Holon	Dashboard Server	Dashboard
Start Node	x						
Order Holon	1	x	1		1		
Sensor Resource Holon	1	1	x	1			
Vibration Sensor & Threshold				x			
Dashboard Resource Holon	1	1			x		
Dashboard Server					1	x	
Dashboard						1	x

Design Structure Matrix PROSA/Erlang Add component	Start Node	Order Holon	Sensor Resource Holon	Vibration Sensor & Noise Filter	Dashboard Resource Holon	Dashboard Server	Dashboard & Graph
Start Node	x						
Order Holon	1	x	1		1		
Sensor Resource Holon	1	1	x	1			
Vibration Sensor & Noise Filter				x			
Dashboard Resource Holon	1	1			x		
Dashboard Server					1	x	
Dashboard & Graph						1	x

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

Design Structure Matrix PROSA/Erlang Replace virt. resource	Start Node	Order Holon	Sensor Resource Holon	Vibration Sensor	GUI Resource Holon	Server	GUI
Start Node	x						
Order Holon	1	x	1		1		
Sensor Resource Holon	1	1	x	1			
Vibration Sensor				x			
GUI Resource Holon	1	1			x		
Server					1	x	
GUI						1	x

Design Structure Matrix PROSA/Erlang Integration	Start Node	Order Holon#1	Scanner Resource Holon	Barcode Scanner	Database#1 Resource Holon	Websocket#1	Server#1	Database#1	UI#1 Resource Holon	UI#1	UI#2 Resource Holon	UI#2	UI#3 Resource Holon	UI#3	*Integration Order Holon*	*Central Database Order Holon*	Order Holon#2	Database#2 Resource Holon	Websocket#2	Server#2	Database#2	Dashboard Resource Holon	Dashboard
Start Node	x																						
Order Holon#1	1	x	1		1				1		1		1		1								
Scanner Resource Holon	1	1	x	1																			
Barcode Scanner				x																			
Database#1 Resource Holon	1	1			x		1																
Websocket#1					1	x																	
Server#1					1		x	1															
Database#1						1	1	x															
UI#1 Resource Holon	1	1							x	1													
UI#1									1	x													
UI#2 Resource Holon	1	1									x	1											
UI#2											1	x											
UI#3 Resource Holon	1	1											x	1									
UI#3													1	x									
Integration Order Holon	1	1													x		1						
Central Database Order Holon	1	1													1	x	1	1					
Order Holon#2	1														1	1	x	1				1	
Database#2 Resource Holon	1																1	x					
Websocket#2																		1	x				
Server#2																		1		x	1		
Database#2																			1	1	x		
Dashboard Resource Holon	1																1					x	1
Dashboard																						1	x

References

1. 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 Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future, 2020.
2. 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 Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future, 2022.
3. 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).
4. Van Brussel, H; Wyns, J; Valckenaers, P; Bongaerts, L; Peeters, P. Reference architecture for holonic manufacturing systems: PROSA *Comput. Ind.* **1998**.
5. Kruger, K; Basson, A. Erlang-based control implementation for a holonic manufacturing cell. *Int. J. Comput. Integr. Manuf.* **2017**.