The Reuse of Business Process Automation Artefacts

Peter A. François[1] and Ralf Plattfaut[2]

Abstract: Reusing business process models and process-related artefacts is deemed an essential factor in BPM initiatives by researchers and practitioners. The reuse of corresponding artefacts has been researched and practiced for almost 40 years. Yet, with the growing adoption of component- and platform-based information systems, a current review of how we view and conduct reuse in Business Process Automation becomes necessary. The advent of citizen developers further increases the need for a consolidated overview of the knowledge on reuse. As citizen developers are not trained to conduct reuse, guidance on how to harness the potential of reuse is needed. Based on a literature review, we identify four main layers on which process automation artefacts are reused and list exemplary technologies and methodologies. In addition, we systematize nine characteristics of reused automation artefacts and nine characteristics that describe a concrete instance of reuse. Finally, we provide a research agenda.

Keywords: Automation Reuse, Business Process Automation, Literature Review, Process Reuse, Reuse Layers, Citizen Developer

1 Introduction

Both supporting digital innovation and enabling quick adaptation of processes and automation solutions are current challenges for business process management (BPM) [Ke21]. There is also wide agreement that – where sensible – business processes and associated artefacts should be reused [Du18, Da15, RV15, RMR15].

While there is broad knowledge about the reuse of business process models (e.g., reference models, reuse of sub-processes, see e.g., [Du18, Ho10, BDK07]), scholarly research has identified the need for more insights on both process reuse [RMR15] and reuse of process automation solutions [As19, Sy20, LW18]. Especially since process automation solutions are considered digital innovations and are said to grow in importance [By15].

Digital innovations have disrupted several markets [Vi19]. The ability of Information Technology (IT) to automate current processes can be a deciding factor in the success of companies [WH20]. Recent disruptions in supply chains and operations have forced entire industries to rapidly adapt their processes or even business models. This pressure also led to the need to adapt corresponding Information Systems (IS) [CCT22, Rö22]. However, companies are partially hindered in building adequate automation solutions themselves

[1] South Westphalia University of Applied Sciences, Lübecker Ring 2, 59494 Soest, francois.peter@fh-swf.de, https://orcid.org/0000-0003-1537-1026
[2] University of Duisburg-Essen, Chair of Information Systems and Transformation Management, Universitätsstraße 2 45141 Essen, ralf.plattfaut@icb.uni-due.de, https://orcid.org/0000-0002-1442-4758

due to a shortage of competent IT workforce. This shortage has persisted almost consistently since the 1980s, leading to a decreased ability to create new IT artefacts and maintain existing ones [Ko84, KS98, NR06, Di15, BNK22].

The reuse of IT artefacts for process automation has been named one measure to cope with both the shortage of competent IT workforce and the need for innovation through quick creation and adaptation of IS (see, e.g., [Ha99, SRK12, Ry19, RJS17]). Next to resolving this dilemma, reuse also brings several benefits, such as improved maintainability, quality, flexibility, productivity and the ability to scale (see, e.g., [LW18, AP10, ZT05, Wi06, Ap90, RJS17, As19]). Especially in process automation, the need for reuse has been highlighted in the literature. For example Lacity and Willcocks [LW18, LW21] describe the reuse of automation artefacts as necessary to be able to reach a significant scale under consideration of cost and time. Dumas et al. [Du18] describe "standardization and reuse across projects" as criteria for a high BPM Maturity.

Since the reuse of artefacts brings such a wide range of benefits, we argue that there is merit in further exploring the reuse of automation artefacts. The ongoing relevance and extensive history of reuse within research (c.f. [Mc68, HHL22]) lend themselves to a literature review. Without clear knowledge and guidelines on automation reuse in the entire artefact lifecycle, several issues arise. Such issues include the inability to design and create IT artefacts in a way that makes them easy to reuse, yet effective when reused (e.g., [ZT05, Ap90, 422, DUB11]), realizing the highest possible value using reusable artefacts using the least resources (see, e.g., [LW18, Wi06, LMS07]) and systematically applying reuse across different hierarchical layers of automation reuse (see, e.g., [KS98]). In this article we therefore aim to uncover the following research questions:

RQ 1. How is reuse conducted in Business Process Automation?
RQ 2. Which Characteristics of artifacts and reuse instances (concrete cases of reuse) are mentioned in the literature?

To further our understanding of the reuse of automation components and to enable BPM and process automation research to make use of related concepts, we first identify layers reuse can be conducted on and then develop characteristics of both the Reuse Artefacts (RAs) and the Reuse Instances (RIs).

2 Background

Following existing literature, we define the reuse of automation solutions as using the same IT-artefact in new circumstances or taking (parts of) an existing artefact [KLJ18, Br01] in order to save on effort when creating or extending an artefact [KS98, Br01, FK05]. The initial artefact may have been previously developed in-house [KS98] or supplied externally [KLJ18].

As such, the reuse of process automation solutions is a cognitive process that involves analysis of the existing artefact in light of new requirements, selecting suitable parts of the

existing artefact, and (if necessary) adapting these parts or adding additional elements (other existing artefacts or specifically developed ones) (see, e.g., [PB13, EJP21, PSH03, As19]) to enable additional automation. Each complete execution of such a process is referred to as a reuse instance (see e.g., [JRS06, RJS17, BB91]).

There has been a long history of reusing automation artefacts. Montecinos [Mo96] argues that humans have always utilized reuse in their reasoning processes and that experience is nothing other than the reuse of existing knowledge and methods. The reuse of information artefacts originated in the 1960s (see, e.g., [Ko07, Mo96, Mc68]). Reuse research first focused on in-house code reuse [RJS17, AP10]. Newer concepts tend to be component-based and work by integrating smaller subsystems (e.g., accessed through such APIs [HHL22]), focusing on interoperability between different systems and languages as well as swift re-configurability and reusability (see, e.g., [Bl00, SJ03, SR05, ZS02]). The ability to quickly adapt systems is especially important when processes change [RL08]. At the same time, some of the "old" concepts of reuse, like functions [KS98] and objects [SJ03] have become an essential part of artefact development. However, a structured overview regarding the concepts of reuse in automation is still missing, which we will provide in the following.

3 Method

As the reuse of artefacts has a long history in IS research and practice, we conducted a structured literature review to answer our research questions. Literature reviews allow for gathering and systemizing rigorously published knowledge within a domain [Pa15]. We followed a three-step sequential process [WW02, Vo15]:

We first applied a keyword-based approach to identify relevant literature and give a comprehensive overview of the concepts used. We used automation reuse (as two words without operators) as keywords across all search fields to first identify relevant articles. As research outlets, we chose the databases of key journals and IS conferences [Vo15]. To uncover BPM-specific concepts, we included the BPM conference. In order to achieve a broad view of our field, we included the AIS e-library as well as the senior scholar basket of eight (EJIS, ISJ, ISR, JIT, JMIS, JSIS, JAIS, MISQ). These outlets were chosen since they represent a broad picture of our field, combining both high scientific relevance (ad established journals) and state-of-the-art publications (ad conferences) [GMT14]. We did not limit the publications reviewed to a specific timespan since older reuse concepts may still be relevant or transferrable to current contexts.

This initial search led to 2075 identified articles. After removing 76 duplicates, the first author manually screened the remaining articles by title for their potential relevance to the topic and research questions. Articles where the title implied that the articles were out of the scope of this review (e.g. "Driver Trust in Automated Driving Systems"). 1.745 articles were excluded based on this. For the remaining 254 articles, the first author manually

screened the abstract and excluded 118 articles mentioning the reuse of software and automation only in passing. This resulted in 136 articles for our thorough analysis. During this analysis, we added another 34 articles through a selective backward search, yielding a total of 170 articles. A foreword search was not deemed necessary due to the broad search term and the inclusion of conference proceedings (see Figure 1).

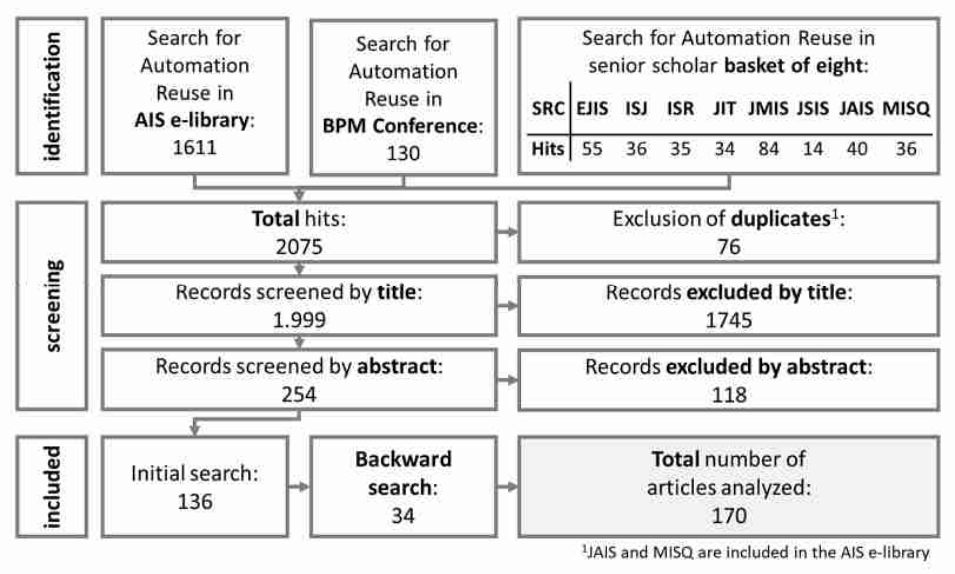

Fig. 1. Literature search process

In the second step, we followed the principles of grounded theory to analyze the literature [WFW13, Ba15]. All relevant articles were initially coded using open coding by both involved researchers to uncover the fundamentals of automation reuse and further relevant themes and theories. Next, we reviewed the resulting codes and aligned similar codes in an iterative process [Pe19] (e.g. "SoftwareComponents_ChoosingGranularity" or "F_Reuse_Layer"). We yielded 89 of these aggregated codes, found in 826 coded segments in the sources.

As a third step, we applied another final coding iteration, in which we applied axial coding within the codes formed in Step 2 to uncover relevant features of automation reuse in a similar iterative process [WFW13]. This second round of axial coding was equivalent to the categories mentioned in our main findings, as depicted in Figure 2 and Table 1. Finally, we compare the results of our grounded-theory-based literature review with the results of prior overview articles (i.e., [KS98, PCH93]).

4 Findings

4.1 Layers of Automation Reuse

Based on our analysis of 170 unique research articles, we extend the work of Poulin et al. [PCH93] and identify four main layers on which automation reuse can occur: Knowledge, Specifications, Software and Architecture (see Figure 2). Here, reuse on a higher layer also includes some degree of reuse on lower layers (e.g., reuse of specifications includes reuse of the corresponding knowledge).

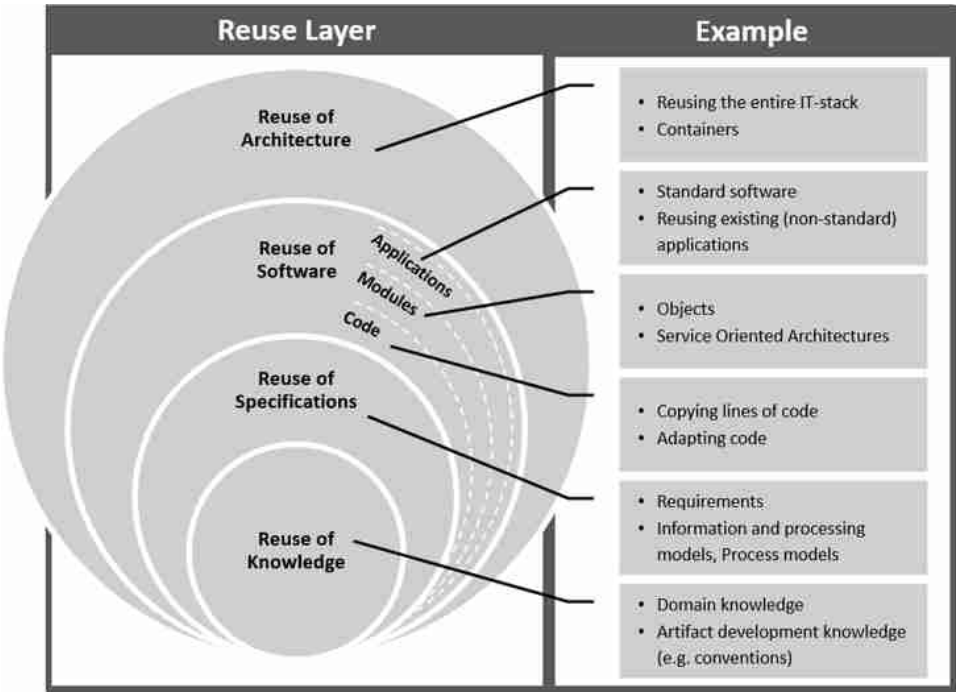

Fig. 2. Layers of Automation Reuse and corresponding examples

The innermost layer identified is the layer of knowledge: Actors can reuse knowledge gained in previous projects for new development [JT08, Ko07, EJP21, Mo96, SM04]. This knowledge includes, for example; domain knowledge [JT08, PSH03, Mo96], the results of data analysis [Ap90], and knowledge on the artefact development processes itself [JT08, FK05, Ap90, EJP21, SM04]. The knowledge reused may be obvious to the actors or hidden in higher layers [SR05, JT08]. One example of structured knowledge reuse are patterns. Patterns offer possible paths to a solution when a specified (often occurring) problem is faced. The knowledge of how to solve such a problem can therefore be reused [Ha99, ACL09, EJP21]. Patterns can also be used together artefacts on other layers such as software [EJP21, Ha99].

In the second layer, knowledge that has been explicated in the form of specifications can be reused [PSH03, ACL09, Ha99, KS98, Da15]. This includes requirements [ACL09, BB91, ZS02], conceptual information/data models [KS98, HAG08], process models or process descriptions (see e.g., [Ap90, MPR20, KS98, Bē15, VS17, ACL09, BDK07]), graphical design [BB91, Ko07], and software documentations [BB91, KS98, Ra99, ACL09]. These specifications may be rigid and (partially) reused as is (see e.g., Fr14, JW08, SD16) or created with the possibility of flexibility / adaptability / configurability, allowing easy reuse in various environments [Fr14, MPR20, BDK07, va13]. Reference processes are another significant concept allowing the reuse of generalized processes across several organizations (see e.g., [Kl12, BDK07, Zh08a, HSV09]).

In the third layer, software created based on the specifications can be reused. Here, the literature differentiates between the reuse of source code, modules, or whole applications. With regards to source code [KS98, Ry19, SM04, Ap90], scholars highlight that single lines of codes [KS98, SM04] or code fragments [KS98] can be reused.

A comparably large research stream focuses on using modules or components (see, e.g., [AP09, Bl00, GRB17]). We understand a module as "a physical packaging of executable software with a well-defined and published interface" [Ho00]. The literature describes modules for use within one technology or system (e.g., one programming language run on one machine) and for use across several. During artefact creation in modern languages, classes or objects [Ba05, Br01, KS98, SJ03, SR05, Wi06, FH00], methods, functions and subroutines [Ry19, KS98, Wi06], as well as standard libraries or packages [KS98] are routinely used (see, e.g., [Ba05]). These artefacts are commonly reused within the same artefact generation process but can also be reused later on (see, e.g., [FH00, SR05]). Such practices have been verified in practice [Sw91] with companies developing repositories for module-based automation reuse [LW18].

Technologies for module (re)use across multiple technologies and systems have gained traction more recently (see, e.g., [SJ03, SR05]). Related technological and organizational questions are still under research (see, e.g., [SWK22, RJS17, Li21]) and several technologies and methodologies in this vein have been proposed [Ha99, Bl00, JRS06, JW07]. Stemming from object-oriented development, concepts such as Component-Based Development [SR05, ZS02], Microservices [Li21, KLJ18] and Services based development (see, e.g., [Li21, RL08, Zh08b]) allow independent components to be freely arranged and thus follow business processes closely. Such services are often designed for reuse or have reuse in mind at the development stage [RL08, KS98]. It may therefore be hard or even impossible to distinguish between the "use" and "reuse" of such components. Kleinert et al. [Kl12] combine the concept of reference processes with reference automations for these processes under the name of "service blueprints". Increasingly there are platforms that allow creating and sharing of modules, enabling users to connect different components created by themselves, the platform owner or third-party suppliers to create information systems from loosely coupled modules [VK19, HHZ18].

Moreover, the era of the reuse of whole applications came with the advent of packaged software applications in the 1960s. Standard software created by software vendors can be

bought and reused in the organization, taking advantage of the proven "standard" functionalities built into them (see, e.g., [RK00, MPR20]). Software-as-a-Service approaches are another way of reusing entire or partial applications (see, e.g., [KLJ18, WOK18]). Next to this, organizations can also reuse their existing artefacts. Schreieck et al., for example, describe using an existing application as part of a new information artefact [SWK22]. In this RI, the legacy system provides the core functionality for a platform ecosystem, allowing (from the platform component creators' viewpoint) the reuse of this core functionality of the system.

In the fourth and final layer, whole IT architectures can be reused. One example of this is containerization, where every system needed for a specific task is bundled into one executable environment [Li21]. In addition, Schreieck et al. describe reusing parts of an IT infrastructure across multiple associated organizations [SWK22]. Karhu et al. describe competitors reusing open-source IT architectures [KGL18].

4.2 Characteristics of Reuse Artefacts and Reuse Instances

The literature describes several characteristics as relevant for automation reuse. The characteristics of RAs (artefact characteristic A1-9) and the characteristics of the specific RI concerning a specific artefact (instance characteristics I1-9) are systemized in Table 1. While some of these characteristics may be relevant when using any (predefined or ad hoc) reuse process on any reuse layer (like described in 4.1) or any combination of reuse layers, some may only be meaningful (or possible) in specific manifestation compositions.

		Characteristic	Manifestation							
Reuse *artefact* characteristics	Artefact creation	(A1) Artefact Layer	Knowledge	Specifications	Software			Architecture		
					Code	Module	Application			
		(A2) Size	Small		Medium		Large			
		(A3) Reuse Orientation	Designed for reuse		Not designed for reuse					
					Adapted for reuse		Not adapted for reuse			
		(A4) Specificity	Customized		Function specific		Generic			
		(A5) Dependency	Independent		Interconnected		Dependent			
	Preparation for Reuse	(A6) Intended approach	Not intended		Bottom-up		Top-down			
		(A7) Visibility	Black box		White box					
		(A8) Documentation	No documentation		Interface documentation		Full documentation			
		(A9) Reuse methodology	Model-based method	Mixed method	Component-based method	Configur. based method	Not method based			

Characteristic (contd.)			Manifestation (contd.)					
Reuse *instance* characteristics	This instance	(I1) Reuse Layer	Knowledge	Specifications	Software			Architecture
					Code	Module	Application	
		(I2) Reuse volume	Artefact partially reused		Artefact mostly reused		Artefact fully reused	
		(I3) Adaptation	Significant		Some		None	
	Environment	(I4) Process Logic	Identical		Similar		Different	
		(I5) Project affiliation	Different project				Same project	
	Reuse methodology	(I6) Reuse approach	Bottom-up				Top-down	
		(I7) Reuse methodology	Model-based method	Mixed method	Component-based method		Configur. based method	Not method based
		(I8) Class of Actor	Original developers	Other developers	Other organizations		Private users	Malicious entities
		(I9) Software Support	Software supported				Not software supported	

Tab. 1: Characteristics of Reuse Artefacts and Reuse Instances

First, we already established that artefacts can be reused on several layers (A1). The reused artefact is associated with a specific layer, yet when the reuse process itself is conducted, it can be reused on the same layer or a different layer (I1). A finished application (e.g., a chatbot automating a customer service process) may be reused as a module in another system (e.g., in a self-service portal), or knowledge explicit in specifications may be reused in another project (e.g., a new chatbot for an internal human resource process).

Another characteristic of RAs is their size (A2). Some artefacts require a certain size in order to be reused. Size is especially important on the component layer. The reuse of large components brings higher value, as more is reused, yet more modification may be required [NR06, SRK12, KS98, PCH93], but small components can be reused on more occasions and can thus produce value more often since they cover less functionality and requirements, yet are harder to find [NR06, RJS17, Ap90, KS98, PCH93]. This trade-off implies the existence of an ideal component size, yet this ideal size is hard to find (see, e.g., [KS98, AP09, Jo09]) and also context-specific [NR06, SRK12].

In the reuse process, the artefact size can also affect the reuse volume (I2). Artefacts can be fully reused, mostly reused (with minor alterations), or only partially reused. Artefacts in the right context and the right size can be fully reused. In other cases, adaptation (I3) is needed (see, e.g., [KS98]). This adaptation may be necessary from a technological perspective since the process logic (I4) differs in the new usage environment.

The amount of adaptation required for reuse depends on the reuse orientation (A3). Here the literature describes artefacts designed for reuse or not designed for reuse (see e. g, [KS98, Ap90, SM04]). When designing an artefact for reuse, the developers had reuse in

mind during development. Ideally, this would include reusability on each layer presented above. This can include artefacts that have been designed for the sole purpose of existing as reusable automation components (see, e.g., [SM04]) or artefacts that have been developed for a specific purpose but with reuse in mind (see, e.g., [ACL09, KS98]).

If the artefact has not been developed with reuse in mind, it can still be adapted for future reuse through some adaptation process (see, e.g., [Br01, AP10, Bē15]). This can happen as part of the specific RI (in I3; see, e.g., [SWK22]) or with the goal of making the artefact reusable in general to potentially be reused at a later stage (e. g, the discovery of relevant legacy artefacts like described in [ACL09]). Also, components not designed for reuse and not adapted for reuse can still be reused using the software (of even single lines of codes) as it is.

According to Kim and Stohr, artefacts have a specificity (A4); They can be customized, functional or generic [KS98]. Customized artefacts are intended to satisfy specific use cases in specific processes according to specific requirements. Functional artefacts are more general in that they are reusable multiple processes within a specific application area or function, such as finance. Generic artefacts can be applied in any field and process where their functionality is required (e.g., a printing operation) [KS98].

Montecinos describes several dependency levels (A5) between artefacts [Mo96]. Independent artifacts can be reused without considering other artifacts. Interconnected artifacts rely on other artifacts and their characteristics and functionality may need to be considered during the reuse process. Dependent artifacts are closely intertwined with the RA and may require major rework for reuse.

Another factor in the reuse process is project affiliation (I5). Here we differentiate between reusing artefacts in the same or a different project [KS98, FH00, SR05]. Reuse in the same project may occur when creating different functionalities or across different lifecycle stages (e.g., reusing knowledge, documentation or existing code during maintenance) [Ap90, KS98, LMS07]. Similarly, artefacts can be reused in a different project.

The reuse approach (I6) can either be top-down or bottom-up. In a top-down approach, artefact creators try to predict the artefacts necessary and create those with the intent of reuse at a later (but not predetermined) time (see, e.g., [SM04]). In a bottom-up approach, existing artefacts are automatically or manually screened to identify potentially reusable components or to extract specific functionalities for specific use cases or later reuse (see, e.g., [DMC21]).

Artefacts can be reused with or without visibility (A7) regarding the inside workings. In a white box reuse case, the artefact is completely accessible to the reusing agent [KS98, SRK12]. In black box reuse, the reusing party does not have insights into the workings of the artefact but can “only” reuse the artefact by interacting with some form of interface and only knows the information given by the artefact developer [Mc68, JT08, JRS06], especially the documentation (see below). Of course, black box reuse may prevent adaptability and partial reuse of the artefact (see, e.g., [JRS06, SRK12]).

Documentation (A8) can be an essential factor in deciding if and how to reuse an automation artefact [Ra99, JT08, Ap90] since documentation increases "understandability" [KS98], trust [Ry19, RL08] and discoverability [DUB11]. In some instances, no documentation may be available for a specific component. In other instances, interface documentation [FSW00] or full documentation [KS98] may be available.

According to Han [Ha99], reuse can be based on different reused methodologies. We have divided this into the intended reuse methodology (A9) and the applied reuse methodology (I7). The literature names five different reuse methodologies [Bē15, Ha99]. First, the component-based methodology relies on assembling components to create larger systems [Ha99]. The model-based methodology uses (and adapts) meta-models to create new artefacts [Ha99]. Model-based and component-based methods can be mixed [Ha99]. Bērziša et al. describe that it is also possible to create RAs under a configuration-based method, where artefacts have built-in decision points to be usable in several contexts [Bē15, BDK07]. In addition, we argue that it is possible to reuse automation not method based (for example, ad hoc copying of single lines of code). The intended reuse methodology can be adhered to in the reuse process or omitted – hence the distinguished applied reuse methodology (I7).

Reuse can be carried out by several classes of actors (I8). These can be the original developers, other developers in the same company, or developers in other organizations [KS98]. Moreover, private persons and malicious entities may reuse artefacts [So14, Ry19, KGL18].

Finally, we found software support (I9) to be one of the important aspects in designing the reuse process (see, e.g., [Ra99, As19]), since the cost of finding out if there is a suitable artefact and where it is located can offset the benefit of reuse [Ra99, KS98, VS17]. Software-supported reuse processes can, for example, offer such a digital repository of RAs (see, e.g., [As19, Ko07, VS17, KS98]), software to extract RAs from legacy code (see, e.g., [ACL09]), or software recommending RAs while developing (see, e.g, [Wa14]).

5 Discussion

Our paper has three main contributions to existing theory. First, we showed the different layers on which automation artefacts can be reused. Here, we showed relevant technologies and concepts used on each layer. Secondly, we collected and systematized nine characteristics regarding RAs and which manifestations of these are mentioned in the literature. Finally, we found and systematized nine characteristics regarding RIs. We discussed the characteristics of RAs and the manifestations described in the literature.

The knowledge of the layers, the characteristics of RAs and RIs can help researchers to understand reusable artefacts better (what are the characteristics of a specific artefact), RIs (how was reuse conducted in this specific act of reuse) and existing reuse process guidelines (which combination of characteristics do the guidelines focus on / support / prohibit).

By defining the layers and characteristics, we enable researchers and practitioners to further advance the topic of automation reuse under a unified, common vocabulary. In addition, new research on reuse process directives can consider the individual layers and characteristics. The characteristics in Table 1 could be combined in various ways to develop previously unknown procedures for reuse support.

By systematizing the layers and characteristics as well as showing examples of relevant technologies, we have shown that there are many different types of automation reuse. Research on reuse in information systems, especially in BPM, should, instead of emphasizing the usefulness and need for reuse of automation artefacts, find out how reuse can best be realized in business process management and automation. We, therefore, (further down in this chapter) formulate a corresponding research agenda.

In addition, our paper holds several implications for practice. According to Rothenberger and Kulkarni, knowledge about the artefact is required to retrieve and find the right component [RK00]. The characteristics of RAs and instances discovered in this paper can be used to describe RAs (e.g., when used in a repository of RAs) by researchers and practitioners. They can additionally help discover and evaluate suitable RAs (e.g., in artefact databases). The overview can also help practitioners decide which layer(s) to perform reuse on and plan for reuse on appropriate layers when generating automation artefacts. When developers design and implement automation artefacts, they can – using the systematized layers and characteristics – be aware of and plan for the reusability of different components of their artefacts (and the artefacts on lower layers). Furthermore, they can use the characteristics mentioned in Table 1 (especially those in the section "Preparation for Reuse") to pave the way for easy reusability.

Still, this paper has three main limitations. Firstly, since it is a pure literature review, we do not know which concepts are (still) relevant in practice. Secondly, adjacent fields have only been covered through the backward search. Finally – since the literature search and paper selection has been conducted by one author – there may be selection bias.

6 Research agenda

Based on our literature review, we derived a research agenda on the reuse of process automation technology. These questions originate from gaps identified in the literature.

First we would recommend research on which combination of characteristics (of the RA, RI, new artefact or additional elements) leads to successful reuse. In this paper we have defined several characteristics of RAs and RIs. Further research should advance these findings and validate them in practice. In addition, real-world data on the influence of characteristics combinations in reuse projects could be used to extend, evaluate and advance reuse process guidelines. Here, it is possible that multiple equifinal combinations of characteristics exist.

Furthermore, we need to find out how the knowledge regime used in developing the initial artefact or in the reuse process influences the success of reuse. Bygstad differentiates between heavyweight and lightweight IT knowledge regimes [By17]. In heavyweight IT, development is "driven by IT professionals, enabled by systematic specification and proven digital technology, and realized through software engineering" [By17]. In lightweight IT, users are enabled to fulfil their own IT needs and generate IT innovation through "cheap and easy-to-use" [By17] systems. Since development under such lightweight knowledge regimes is end-user-driven, users will have varying IT skill levels and knowledge about reuse. Well-designed support for reuse could help harness its potential in this area as well.

Item	Question(s)
Overarching	Which combination of characteristics leads to successful reuse?
	When should organizations rely on the reuse of process automation artefacts? Which environmental factors or characteristics of the components make reuse viable?
	How does the knowledge regime used in developing the initial artefact or in the reuse process influence the success of reuse?
	What can automation reuse research learn from adjacent streams such as knowledge management or requirements engineering?
	Since automation reuse stems from reuse in manufacturing in the 1960s (see above) are there newer concepts in manufacturing that could again be transferred for reuse?
	Which types of reuse are mutually beneficial, and which are obstructive / incompatible?
Reuse Layer	How to best combine / stack the different layers, technologies, and concepts of automation reuse in Business Process Automation?
Artefact	How to methodically decide which parts of an artefact to reuse?
Reuse Process	Which of the existing reuse process guidelines support reuse with which configuration of characteristics well?
	Which new reuse process guidelines could be developed to improve additional configurations?
Additional development	Which degree of additional development (also in the sense of requirements engineering and testing) should be performed during reuse?
New Artefact	How do properties of the reuse process or properties of the reused artefact affect the new artefact (short or long term)?
	How can artefacts developed with reuse be reused and what does re-reuse entail?

Table 2. Research agenda

Moreover, it would be valuable to evaluate further when organizations should rely on the reuse of process automation artefacts. While several proposals for the structure of reuse processes itself exist (see, e.g., [HAG08, ACL09, Sw91, PSH03, HHL22, Bl00]), there seem to be few up-to-date articles focusing on concrete instructions on when to reuse (e.g.,

[CB91]). The general consensus seems to be, that reuse should be conducted when it is financially beneficial [KS98, RK00], leads to better quality or increases development speed [CB91]. However, we argue that these characteristics are complex for programmers to evaluate when "out in the field". This issue increases when many possible reuse sources are available (e.g., large codebase, many components in a platform ecosystem) [RK00, NR06]. Therefore, we argue that research should be done on how these characteristics can be determined algorithmically to guide developers.

We also call for more research on how to best combine or stack the different layers of automation reuse and combine the technologies and concepts developed for automation reuse. While some concepts (e.g., model-driven development [LMS07]) rely explicitly on utilizing different layers to enable reuse, we have not seen any guidelines on how to systematically reuse artefacts across all layers in different contexts.

Another interesting problem is the degree of additional development that should be accepted during reuse (in regular or lightweight programming), like functional changes, such as adapting or expanding functionality, (additional) requirements engineering and testing for the new environment [KS98, SRK12, Ra99, PB13].

In addition, very little work seems to be done on the workings of the artefacts created through reuse. Baskerville et al. describe a case study where adaptations to the new artefact did not negatively affect the new artefact [Ba05]. However, it is unclear what the long-term effects of reuse are, which configurations of artefact characteristics, reuse process and additional elements lead to successful new artefact creation. It is also unclear how these new artefacts can be re-reused and its implications.

7 Conclusion

One of the basic rules in BPM is that the reuse of process models should take precedence before the new development [Du18, RMR15, RV15]. This precedence for reuse also extends to automation artefacts (Davenport, for example, states this for the knowledge layer [Da15]). The need for research on reusing automation solutions has been well described [As19, Sy20, LW18].

In this paper, we have demonstrated the state of the reuse of process automation artefacts in the IS literature. We have shown and systematized layers of automation reuse, related technologies, the characteristics of RAs, and the reuse process.

There is still much work for researchers in the area of automation reuse, as seen in the proposed research agenda. In 1968, McIlroy already called for adapting principles and technologies from the field of mechanical engineering for use in software engineering [Mc68]. He called for "manufacturers of standard parts", which we now see in the existence of standard software and external service vendors, and for the ability to "order parts to individual specifications of size, ruggedness, speed, capacity, precision or character set" which we now see in requirements engineering, and "catalogues of standard parts", which

we now see in the form of reuse repositories and platform ecosystems stores. While we now have access to these technologies (and many others as described above) and could utilize them for further reuse, the reuse of automation artefacts still seems to pose hurdles for researchers and practitioners. Thus, we hope our findings can help to advance the understanding of reuse and to tap into the potential that reuse holds for BPM and Business Process Automation.

Acknowledgements

This work has been partially developed in the project KEBAP at South Westphalia University of Applied Sciences. The project (reference number: 13FH034KX0) is partly funded by the German Ministry of Education and Research (BMBF).

References

[ACL09] Aldin, L.; Cesare, S. de; Lycett, M.: Semantic Discovery And Reuse Of Business Process Patterns. MCIS 2009 Proceedings, 2009.

[AP09] Antikainen, J.; Pekkola, S.: Factors influencing the alignment of SOA development with business objectives. ECIS 2008 Proceedings, 2009.

[AP10] Allen, G.; Parsons, J.: Is Query Reuse Potentially Harmful? Anchoring and Adjustment in Adapting Existing Database Queries. Information Systems Research 1/21, p. 56–77, 2010.

[Ap90] Apte, U. et al.: Reusability-Based Strategy for Development of Information Systems: Implementation Experience of a Bank. MIS Quarterly 4/14, p. 421–433, 1990.

[As19] Asatiani, A. et al.: Organizational Implementation of Intelligent Automation as Distributed Cognition: Six Recommendations for Managers. ICIS 2019 Proceedings, 2019.

[Ba05] Baskerville, R. et al.: Extensible Architectures: The Strategic Value of Service Oriented Architecture in Banking. ECIS 2005 Proceedings, 2005.

[Ba15] Bandara, W. et al.: Achieving Rigor in Literature Reviews: Insights from Qualitative Data Analysis and Tool-Support. Communications of the Association for Information Systems 37, 2015.

[BB91] Barnes, B. H.; Bollinger, T. B.: Making reuse cost-effective. IEEE Software 1/8, p. 13–24, 1991.

[BDK07] Becker, J.; Delfmann, P.; Knackstedt, R.: Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In (Becker, J.; Delfmann, P. Hrsg.): Reference Modeling. Efficient Information Systems Design Through Reuse of Information Models. Physica, Heidelberg, p. 27–58, 2007.

[Bē15] Bērziša, S. et al.: Capability Driven Development: An Approach to Designing Digital Enterprises. Business & Information Systems Engineering 1/57, p. 15–25, 2015.

[Bl00] Blake, M. B.: A Development Approach for Workflow-Based E-Commerce using Reusable Distributed Components. AMCIS 2000 Proceedings, 2000.

[BNK22] Bossler, L. F.; Noeltner, M.; Kroenung, J. S.: Attitude Discrepancy and Its Influence on Turnover Intention Among IS Professionals. ICIS 2022 Proceedings 15, 2022.

[Br01] Brash, D.: Reuse in Information Systems Development: Classification and Comparison. ACIS 2001 Proceedings., 2001.

[By15] Bygstad, B.: The Coming of Lightweight IT. ECIS 2015 Completed Research Papers, 2015.

[By17] Bygstad, B.: Generative Innovation: A Comparison of Lightweight and Heavyweight IT. JIT (Journal of Information Technology) 2/32, p. 180–193, 2017.

[CB91] Caldiera, G.; Basili, V. R.: Identifying and qualifying reusable software components. Computer 2/24, p. 61–70, 1991.

[CCT22] Compeau, D.; Correia, J.; Thatcher, J.: When Constructs Become Obsolete: A Systematic Approach to Evaluating and Updating Constructs for Information Systems Research. MIS Quarterly 2/46, p. 679–712, 2022.

[Da15] Davenport, T. H.: Process Management for Knowledge Work. In (Vom Brocke, J.; Rosemann, M. Hrsg.): Handbook on Business Process Management 1. Introduction, Methods, and Information Systems. Springer, Berlin, Heidelberg, p. 17–35, 2015.

[Di15] Dinger, M. et al.: Does Professionalism Matter in the IT Workforce? An Empirical Examination of IT Professionals. Journal of the Association for Information Systems 4/16, 2015.

[DMC21] Diaz, O.; Medina, H.; Contell Jeremias P.: Promoting Design Knowledge Accumulation Through Systematic Reuse. HICCS 2021 Proceedings, 2021.

[Du18] Dumas, M. et al.: Fundamentals of Business Process Management. Springer, Berlin Heidelberg, 2018.

[DUB11] Dudek, S.; Uebernickel, F.; Brenner, W.: Towards Computer Aided It Service Engineering. MCIS 2011 Proceedings, 2011.

[EJP21] Ebel, M.; Jaspert, D.; Poeppelbuss, J.: Pattern-Based Smart Service Innovation. PACIS 2021 Proceedings, 2021.

[FH00] Ferguson, R. I.; Hunter, A.: On the Use of Object Oriented Techniques to Support the Construction of CASE Tools. AMCIS 2000 Proceedings, 2000.

[FK05] Frakes, W. B.; Kang, K.: Software Reuse Research: Status and Future. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 7/31, p. 529–536, 2005.

[Fr14] Frank, U. et al.: The Research Field “Modeling Business Information Systems”. Business & Information Systems Engineering 1/6, p. 39–43, 2014.

[FSW00] Fan, M.; Stallaert, J.; Whinston, A. B.: The adoption and design methodologies of component-based enterprise systems. EJIS (European Journal of Information Systems) 1/9, p. 25–35, 2000.

[GMT14] Gogan, J. L.; McLaughlin, M.-D.; Thomas, D.: Critical Incident Technique in the Basket. ICIS 2014 Proceedings, 2014.

[GRB17] Gruettner, A.; Richter, J.; Basten, D.: Explaining The Role Of Service-Oriented Architecture For Cyber-Physical Systems By Establishing Logical Links. ECIS 2017 Proceedings, 2017.

[Ha99] Han, T.-D.: Automating Reuse for Systems Design. AMCIS 1999 Proceedings, 1999.

[HAG08] Hammoudi, S.; Alouini, W.; Gammoudi, M. M.: On The Challenge Of A Semi-Automatic Transformation Process In Model Driven Enterprise Information Systems. MCIS 2008 Proceedings, 2008.

[HHL22] Huang, J. C.; Henfridsson, O.; Liu, M. J.: Extending digital ventures through templating. Information Systems Research 1/33, p. 285–310, 2022.

[HHZ18] Hahn, C.; Huntgeburth, J.; Zarnekow, R.: Leverage Once, Earn Repeatedly—Capabilities for Creating and Appropriating Value in Cloud Platform Ecosystems. In (Lamboglia, R. et al. Hrsg.): Network, Smart and Open. Springer International Publishing, Cham, p. 143–164, 2018.

[Ho00] Hopkins, J.: Component primer. Communications of the ACM 10/43, p. 27–30, 2000.

[Ho10] Hofstede, A. H. M. et al.: Modern Business Process Automation. Springer, Berlin, Heidelberg, 2010.

[HSV09] Hoyer, V.; Stanoevska-Slabeva, K.; Vom Brocke, J.: On the Contribution of Reference Modeling for Organizing Enterprise Mashup Environments. In (Rinderle-Ma, S.; Sadiq, S.; Leymann, F. Hrsg.): LNBIP 43. Business Process Management Workshops, 2009.

[Jo09] Joachim, N. et al.: Examining the Organizational Decision to Adopt Service-Oriented Architecture (SOA) - Development of a Research Model. DIGIT 2009 Proceedings, 2009.

[JRS06] Jain, H.; Rothenberger, M.; Sugumaran, V.: Flexible Software Component Design Using A Product Platform Approach. ICIS 2006 Proceedings, 2006.

[JT08] Juhrisch, M.; Thies, G.: Service Management Beyond UDDI - A Design Science Approach. PACIS 2008 Proceedings, 2008.

[JW07] Juhrisch, M.; Weller, J.: On the Reuse of SOA Components on Business Process Analysis Stages. PACIS 2007 Proceedings, 2007.

[JW08] Juhrisch, M.; Weller, J.: Connecting Business and IT - A Model-driven Web Service based Approach. PACIS 2008 Proceedings, 2008.

[Ke21] Kerpedzhiev, G. D. et al.: An Exploration into Future Business Process Management Capabilities in View of Digitalization. Business & Information Systems Engineering 2/63, p. 83–96, 2021.

[KGL18] Karhu, K.; Gustafsson, R.; Lyytinen, K.: Exploiting and Defending Open Digital Platforms with Boundary Resources: Android's Five Platform Forks. Information Systems Research 2/29, p. 479–497, 2018.

[Kl12] Kleinert, T. et al.: Systematic Identification of Service-Blueprints for Service-Processes - A Method and Exemplary Application. BPM 2012 Workshops, LNBIP 132, p. 598–610, 2012.

[KLJ18] Krancher, O.; Luther, P.; Jost, M.: Key Affordances of Platform-as-a-Service: Self-Organization and Continuous Feedback. Journal of Management Information Systems 3/35, p. 776–812, 2018.

[Ko07] Kotlarsky, J. et al.: Globally Distributed Component-Based Software Development: an Exploratory Study of Knowledge Management and Work Division. JIT (Journal of Information Technology) 2/22, p. 161–173, 2007.

[Ko84] Konsynski, B. R.: Advances in Information System Design. Journal of Management Information Systems 3/1, p. 5–32, 1984.

[KS98] Kim, Y.; Stohr, E. A.: Software Reuse: Survey and Research Directions. Journal of Management Information Systems 4/14, p. 113–147, 1998.

[Li21] Li, S. et al.: Understanding and Addressing Quality Attributes of Microservices Architecture: A Systematic Literature Review. Inf. Softw. Technol. (Information and Software Technology) 106449/131, 2021.

[LMS07] Lycett, M.; Marcos, E.; Storey, V.: Model-driven systems development: an introduction. European Journal of Information Systems 16, p. 346–348, 2007.

[LW18] Lacity, M.; Willcocks, L. P.: Innovating in Service: The Role and Management of Automation. In (Willcocks, L.; Oshri, I.; Kotlarsky, J. Hrsg.): Dynamic innovation in outsourcing. Theories, cases and practices. Palgrave Macmillan, Cham, 2018.

[LW21] Lacity, M.; Willcocks, L.: Becoming Strategic with Intelligent Automation. MIS Quarterly Executive, p. 169–182, 2021.

[Mc68] McIlroy, M. D.: Mass Produced Software Components. NATO Software Engineering Conference. Garmisch, Germany, 7th to 11th October 1968, 1968.

[Mo96] Montecinos, F.: An experience in the establishment of a software reuse culture in a real environment. AMCIS 1996 Proceedings., 1996.

[MPR20] Mendling, J.; Pentland, B. T.; Recker, J.: Building a complementary agenda for business process management and digital innovation. European Journal of Information Systems 3/29, p. 208–219, 2020.

[NR06] Nazareth, D. L.; Rothenberger, M.: Does the 'Golidlocks Conjecture' Apply to Software Reuse? Journal Of Information Technology Theory And Application 2/8, p. 57–67, 2006.

[Pa15] Paré, G. et al.: Synthesizing information systems knowledge: A typology of literature reviews, Information & Management. Information & Management 2/52, p. 183–199, 2015.

[PB13] Pahlke, I.; Beck, R.: End User Empowerment through Platforms for Situational Applications. All Sprouts Content, 2013.

[PCH93] Poulin, J. S.; Caruso, J. M.; Hancock, D. R.: The business case for software reuse. IBM Systems Journal 4/32, 1993.

[Pe19] Pekkola, S. et al.: The Magical "We": Enhancing Collaboration Transparency in Grounded Theory Method in Information Systems Research. Communications of the Association for Information Systems 45, p. 251–269, 2019.

[PSH03] Purao, S.; Storey, V. C.; Han, T.: Improving Analysis Pattern Reuse in Conceptual Design: Augmenting Automated Processes with Supervised Learning. Information Systems Research 3/14, p. 269–290, 2003.

[Ra99] Ravichandran, T.: Software reusability as synchronous innovation: a test of four theoretical models. European Journal of Information Systems 3/8, p. 183–199, 1999.

[RJS17] Rothenberger, M. A.; Jain, H.; Sugumaran, V.: A Platform-based Design Approach for Flexible Software Components. Journal of Information Technology Theory and Application 2/18, p. 29–56, 2017.

[RK00] Rothenberger, M. A.; Kulkarni, U. R.: Software Reuse in Information Systems Development. AMCIS 2000 Proceedings, 2000.

[RL08] Ren, M.; Lyytinen, K. J.: Building Enterprise Architecture Agility and Sustenance with SOA. Communications of the Association for Information Systems 22, 2008.

[RMR15] Reijers, H. A.; Mendling, J.; Recker, J.: Business Process Quality Management. In (Vom Brocke, J.; Rosemann, M. Hrsg.): Handbook on Business Process Management 1. Introduction, Methods, and Information Systems. Springer, Berlin, Heidelberg, p. 167–185, 2015.

[Rö22] Röglinger, M. et al.: Exogenous Shocks and Business Process Management. A Scholars' Perspective on Challenges and Opportunities. Business & Information Systems Engineering 5/64, p. 669–687, 2022.

[RV15] Rosemann, M.; Vom Brocke, J.: The Six Core Elements of Business Process Management. In (Vom Brocke, J.; Rosemann, M. Hrsg.): Handbook on Business Process Management 1. Introduction, Methods, and Information Systems. Springer, Berlin, Heidelberg, 2015.

[Ry19] Ryan, T. J. et al.: The Influence of Personality on Code Reuse. HICSS 2019 Proceedings, p. 5805–5814, 2019.

[SD16] Saarsen, T.; Dumas, M.: Factors Affecting the Sustained Use of Process Models. In (La Rosa, M.; Loos, P.; Pastor, O. Hrsg.): Business Process Management Forum. BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings. Springer, Cham, 2016.

[SJ03] Satzinger, J. W.; Jackson, R. B.: Making the Transition from OO Analysis to OO Design with the Unified Process. Communications of the Association for Information Systems 12, 2003.

[SM04] Sherif, K.; Menon, N. M.: Managing Technology and Administration Innovations: Four Case Studies on Software Reuse. Journal of the Association for Information Systems 7/5, 2004.

[So14] Sojer, M. et al.: Understanding the Drivers of Unethical Programming Behavior: The Inappropriate Reuse of Internet-Accessible Code. Journal of Management Information Systems 3/31, p. 287–325, 2014.

[SR05] Sharp, J.; Ryan, S.: A Review of Component-Based Software Development ICIS 2005 Proceedings, 2005.

[SRK12] Subramanyam, R.; Ramasubbu, N.; Krishnan, M. S.: In Search of Efficient Flexibility: Effects of Software Component Granularity on Development Effort, Defects, and Customization Effort. Information Systems Research 3-part-1/23, p. 787–803, 2012.

[Sw91] Swanson, K. et al.: The Application Software Factory: Applying Total Quality Techniques to Systems Development. MIS Quarterly 4/15, p. 567–579, 1991.

[SWK22] Schreieck, M.; Wiesche, M.; Krcmar, H.: Governing innovation platforms in multi-business organisations. European Journal of Information Systems, p. 1–22, 2022.

[Sy20] Syed, R. et al.: Robotic Process Automation: Contemporary themes and challenges. Computers in Industry 115, p. 103162, 2020.

[va13] van der Aalst, W. M. P.: Business Process Management: A Comprehensive Survey. ISRN Software Engineering Article ID 507984,/Volume 2013, 2013.

[Vi19] Vial, G.: Understanding digital transformation: A review and a research agenda. The Journal of Strategic Information Systems 2/28, p. 118–144, 2019.

[VK19] Vestues, K.; Knut, R.: Making Digital Infrastructures More Generative Through Platformization and Platform- driven Software Development: An Explorative Case Study. 10th Scandinavian Conference on Information Systems, 2019.

[Vo15] Vom Brocke, J. et al.: Standing on the Shoulders of Giants: Challenges and Recommendations of Literature Search in Information Systems Research. Communications of the Association for Information Systems 37, 2015.

[VS17] Veitch, R.; Seymour, L. F.: Business Process Model Reuse In A Multi-Channel / Multi-Product Environment–Problem Identification And Tentative Design. MCIS 2017 Proceedings, 2017.

[Wa14] Wang, D. et al.: The Design of a Workflow Recommendation System for Workflow as a Service in the Cloud. In (Lohmann, N.; Song, M.; Wohed, P. Hrsg.): Business Process Management Workshops. Springer International Publishing, Cham, p. 251–263, 2014.

[WFW13] Wolfswinkel, J. F.; Furtmueller, E.; Wilderom, C. P. M.: Using grounded theory as a method for rigorously reviewing literature. European Journal of Information Systems 1/22, p. 45–55, 2013.

[WH20] Wiesböck, F.; Hess, T.: Digital innovations. Electron. Markets (Electronic Markets) 1/30, p. 75–86, 2020.

[Wi06] Witman, P. D.: Tracing Patterns of Large-Scale Software Reuse. AMCIS 2006 Proceedings., 2006.

[WOK18] Willcocks, L.; Oshri, I.; Kotlarsky, J. Hrsg.: Dynamic innovation in outsourcing. Theories, cases and practices. Palgrave Macmillan, Cham, 2018.

[WW02] Webster, J.; Watson, R. T.: Analyzing the past to prepare for the future: Writing a literature review. MIS Quarterly 2/26, p. xiii–xxiii, 2002.

[Zh08a] Zhu, L. et al.: Challenges Observed in the Definition of Reference Business Processes. In (Hutchison, D. et al. Hrsg.): Business Process Management Workshops. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 95–107, 2008.

[Zh08b] Zhao, J. L. et al.: Impact of Service-Centric Computing on Business and Education. Communications of the Association for Information Systems 22, 2008.

[ZS02] Zhao, L.; Siau, K.: Component-Based Development Using UML. Communications of the Association for Information Systems 9, 2002.

[ZT05] Zhang, Y.; Tanniru, M.: Business Flexibility and Operational Efficiency - Making Trade-Offs in Service Oriented Architecture. AMCIS 2005 Proceedings, 2005.