Improving Software Flexibility for Business Process Changes

The time between changes in business processes and their IT implementation has an impact on a company's competitiveness. In addition, the costs of such changes should be minimized. The article presents a method for implementing changes to business processes based on a process platform. By means of simulation it is shown that this method offers several advantages compared to traditional component-oriented software development. Changes are implemented with low labor and transaction costs and operational flexibility is increased.

DOI 10.1007/s12599-009-0086-8

The Authors

Dipl.-Ing. Oliver Holschke
Dipl.-Ing. Jannis Rake
Fachgebiet Systemanalyse und EDV
Technische Universität Berlin
Franklinstr. 28-29
10587 Berlin
Germany
oliver.holschke@sysedv.tu-berlin.de
url: http://www.sysedv.tu-berlin.de

Dr. Philipp Offermann
Dr. Udo Bub
Deutsche Telekom AG Laboratories
Ernst-Reuter-Platz 7
10587 Berlin
Germany
philipp.offermann@telekom.de
udo.bub@telekom.de
url: http://www.laboratories.telekom.com

Received: 2009-05-01
Accepted: 2009-10-21
Accepted after two revisions by the editors of the special focus.

This article is also available in German in print and via http://www.wirtschaftsinformatik.de: Holschke O, Rake J, Offermann P, Bub U (2010) Steigerung der Softwareflexibilität bei Geschäftsprozessänderungen. WIRTSCHAFTSINFORMATIK. doi: 10.1007/s11576-009-0210-x.

1 Introduction

Companies must be able to carry out organizational adjustments effectively and efficiently to secure sustainable competitive advantages (Cyert and March 1963). For these necessary adjustments flexibility of business processes constitutes a key influence factor (Moitra and Ganesh 2005; Regev et al. 2007). Information technology (IT) has proved to be a crucial driver of business process flexibility in dynamic business environments (Clemons 1986; King et al. 1989). So far, the potential of IT for the organizational adaptation could not be fully accessed, which regularly appears as a lack of alignment between IT and the organization and/or the business processes – the so-called business-IT gap – (see Masak 2006; Chan and Reich 2007; Aier and Winter 2009). Evidence for this is the difficult individualization of complex operational software systems with regard to the variable requirements of the business processes (Brehm et al. 2001; Hong and Kim 2002).

Business processes are increasingly supported by software- and platform-as-a-service offers (SaaS/PaaS) (Buxmann et al. 2008), for which market sizes of up to 16 billion U.S. dollars have been forecasted (Ried et al. 2009). Successful PaaS providers (such as salesforce.com, NetSuite) succeed in making cross-industry solutions accessible to a broad user group. Therefore, flexibility of the provided software covering the changing requirements plays a crucial role. However, the actual implementation of the changes is always associated with costs, which may vary depending on the underlying architecture.

The following contribution examines whether a deployment of business processes based on a service-oriented platform is able to cope with changes more flexibly and efficiently than a classic component-based approach. The article is structured as follows. To better understand software-based changes in business processes and their evaluation, in Sect. 2 we first present a typology. Subsequently, we discuss the criterion of business process flexibility. Based on these observations we develop a model to simulate and evaluate the expenses for these changes in Sect. 4. In Sect. 5, we present the platform-based method, including a differentiation from the component-oriented approach. Based on selected change scenarios we perform a simulation and evaluation of the platform approach and discuss their results. Finally, the limitations of the model will be outlined. The contribution closes with a conclusion and an outlook on future studies.

2 Software Changes: A Typology

The literature on information systems change (ISC) covers a wide area as organizations using information systems may change in regard to a variety of dimensions, ranging from psychosocial to technological aspects. According to the classification of Lyytinen (1987), ISC is a process of creating and/or configuring elements and connections, and which takes place at and between three areas of IS: (1) symbols; (2) organizational tasks, structures, and processes; and (3) technological core of the organization.

This contribution is limited to the symbolic area, in particular to possible modeling grammars for software modification, and the organizational processes and structures in terms of business processes and their data. Changes in the

technological core of the organization are out of the focus of this contribution. An additional restriction is made on the level of the extent of changes. Since the focus here is on software changes that are required by daily business and with limited amount of adaptations, we address the incremental forms of ISC as distinguished from episodic, radical, and occasional changes (Gersick 1991). Our literature study of typologies for software modification yielded a very broad diversification. Therefore, the following considerations were used to structure the analysis:

- *Grammar type*: Describes the visual representation format, including the rules according to which the user can make a modification.
- *Affected layer(s)*: A generic 3-layer model for application systems with communication layer, application layer, and database layer to which the modifications may relate (Ferstl and Sinz 1998; Brehm et al. 2001).
- *Granularity of the observation*: Ranges from fine (specific guidelines for action for the user, such as the selection of a value in a drop-down menu) to coarse (an IT project as a change activity, such as the launch of an ERP suite).
- *End-user focus (in terms of Web 2.0)*: Describes how the modification activities are aligned to typical end-users (focus on professional competence; predominantly working with word processing, spreadsheet tools, and browser-based applications). Highly user-focused modification options include e.g. the setting and connection of weblogs and online directories (Schroth and Janner 2007).

Table 1 presents the identified modification activities according to authors and the above mentioned structural features. The literature study shows that specific modification activities for various grammar types, layers of granularity, and end-users have been classified. The focus of previous works is set on the areas of software maintenance, ERP customization, and configuration of process models, while in more recent works modifications at the level of comprehensive artifacts, such as sub-processes (Weber et al. 2007) and complete IT projects (Dreyfus and Iyer 2008), are increasingly addressed. Despite the differences in the structural features, comprehensive types of modification activities can be observed, as the intersections of the concepts (in italics and underlined) in the right column of **Table 1** clearly indicate. These activities are: (a) add, (b) delete, (c) move, (d) adjust and (e) create. For the configuration, the activities (f) lock/unlock should be highlighted. Thus, we can identify a core set of modification activities independently of grammar, addressed system layers, and granularity, which can be consulted for a model-based approach of software flexibility.

3 Business Process Flexibility and Efficiency

For a detailed analysis of various software change approaches, quality criteria are required on the basis of which management decisions can be taken. Business process flexibility is seen as an important quality measure for increased performance of companies in volatile markets (Davenport and Short 1990; Allweyer 1998; Berry and Cooper 1999; Kumar 2004; Gebauer and Lee 2008). This contribution relies on the work of Sethi and Sethi (1990), who characterized various facets of flexibility in a classification of 11 types of flexibility. Following this classification, our underlying consideration of the flexibility concept essentially addresses operational flexibility. In classical literature, operational flexibility refers to an object to be manufactured, which can be produced through traversing alternative process paths. In the context of software modification, operational flexibility can be transferred to a specific modification which must be "produced". Alternative options for the realization arise from the fact that further employees – in addition to the person initially involved – now have the capacity to conduct further modifications. For example, specialized domain experts now have the ability to integrate an additional activity in a process model – an operation which has been a programmer's task prior to the capacity shift. Multiple paths for making a specific modification result in higher operational flexibility.

While the above remarks follow a classical notion of flexibility, the implementation of process and software changes does not only involve operational flexibility, i.e. specifically increasing the number of qualified staff members, but also – as mentioned above – the efficiency of the implementation. Efficiency requires an amount of effort, typically time and costs, which is attributed to a well-defined activity, in our case to specific process and software changes. Approaches for software modification must adequately fulfill both criteria – flexibility and efficiency – in order to successfully face the system environment.

4 Simulation and Evaluation Model

Based on the works on flexibility characteristics of work systems we consider an information system to be a work system consisting of *resources* and *requirement types* (Iravani et al. 2005; Alter 2008). This view can also be reconciled with fundamental concepts of queuing and coordination theory (Malone and Crowston 1994). Since we focus on the specific context of changing software systems supporting business processes, we consider resources to be *actors* (the employees involved in the software change process of the organization) and the requirement types to be the respective *modification operations* (see Sect. 2). The actors execute the various modification operations. Dependent on the skills they possess, they can or cannot perform certain operations. Similar to the modification operations, skills can be regarded as capacity or means by which these operations can be implemented. **Fig. 1** illustrates all potential modification operations as a combination of the basic types of modification across the language levels.

The coverage of a set of modification requirements by a set of actors with different capacities/skills can be mapped into a connected graph. **Fig. 2** illustrates two exemplary distributions of skills for N actors (A_N) and K modification operations (M_K). For brevity reasons, the model has the values $N = 3$ and $K = 9$. A solid edge between an actor and a modification requirement represents that the actor has the necessary skills to implement the operational requirement type. A dashed edge denotes an involvement of an actor by means of communication, e.g. through the formulation of requirements.

According to our above given definition, higher operational flexibility exists if a modification can be carried out by several actors; i.e. the number of incoming edges of a modification operation can be taken as an index for operational flexibility. The operational flexibility of the overall system is represented by the vector

Table 1 Process or software modification types identified in literature

Author	Grammar type	Affected layer(s)	Granularity of the observation	End-user focus (in terms of Web2.0)	Modification activities
Hoyer et al. (2008); Schroth and Janner (2007)	S	CL, A, D possible	Fine-medium	Strong	Lightweight composition of applications (enterprise mash-up): including *add* and *delete* data inputs and outputs and control flow dependencies between applications (widgets); *add* and *delete* links between widgets and resources (piping); *create*, *add*, and *delete* widgets; *move* widgets.
Gottschalk et al. (2008); Rosemann and Aalst (2007)	S	CL, A, D possible	Fine	Medium	Configuration of process models. Based on *block* and *hide* process model elements. To provide a flexible, enterprise-specific coverage, the underlying process model must provide comprehensive options.
Weber et al. (2007)	S	CL, A, D possible	Fine	Medium	Proposal of 13 modification types: including *add*, *delete*, *move*, *replace*, *interchange* process fragment; *add*, *delete* control flow dependency.
Allweyer (1998); Remme (1997)	S	CL, A, D possible	Medium	Medium-weak	Modifications at data, control flow, and organizational level: including data porting: *add* attributes; control flow change: *delete*/*add* control flow dependencies; omission or new function: *delete*/*add* a function, *adjust* function; also configuration of functions, i.e. *lock* and *unlock*.
Brehm et al. (2001); See Pui Ng et al. (2002)	T, C	CL, A, D possible	Fine	Weak	Types of ERP modifications: including configuration using tables; *add* third-party application; *create* and *add* masks; programming reporting options → *create* and *add*; *create* and *add* new workflows; patch; upgrade.
Turkay et al. (2004)	S, T	A, D	Fine	Weak	Model-based configuration of middleware. Model-based and alphanumeric setting of optional parameters, i.e. *lock* or *unlock*.
Swanson (1976); Mockus and Votta (2000)	C	A, D	Fine	Weak	Typology of software modification for maintenance purposes: *add* new software features; correct errors; code restructuring → *adjust*.
Dreyfus and Iyer (2008); Baldwin and Clark (2000)	–	A	Coarse	Weak	General modification types of module changes: *connect*, *share*, *add*, *substitute*, *generalize*, *delete*, *port*, *adjust*.

S: Symbol-based grammar, T: Table, C: Code

CL: Communication layer, A: Application layer, D: Database layer

consisting of the individual levels of flexibility. Therefore, the operational flexibility on the left-hand side in **Fig. 2** is (2; 2; 2; 2; 2; 2; 1; 1; 1) and (2; 2; 2; 2; 2; 2; 2; 2; 1) on the right-hand side. Following Iravani et al. (2005), a higher operational flexibility can be assumed for larger values in the vectors. To develop an indicator it is necessary to represent the larger values in the vector. Average determination would be an example for the formation of a possible indicator. Hence, we have an operational flexibility of 15/9 for the system on the left and of 17/9 for the

Language construct (to which the modification is related) / **Modification type**	Activity	Flow	Gate	Event	User interface	Process fragment	Service	Role	Configuration table	Software component	Code
Add	M_1	M_6	M_{11}	M_{16}	M_{21}	M_{26}	M_{31}	M_{36}	M_{41}	M_{46}	M_{51}
Delete	M_2	M_7	M_{12}	M_{17}	M_{22}	M_{27}	M_{32}	M_{37}	M_{42}	M_{47}	M_{52}
Move	M_3	M_8	M_{13}	M_{18}	M_{23}	M_{28}	M_{33}	M_{38}	M_{43}	M_{48}	M_{53}
Adjust	M_4	M_9	M_{14}	M_{19}	M_{24}	M_{29}	M_{34}	M_{39}	M_{44}	M_{49}	M_{54}
Create	M_5	M_{10}	M_{15}	M_{20}	M_{25}	M_{30}	M_{35}	M_{40}	M_{45}	M_{50}	M_{55}

Fig. 1 Modification operations

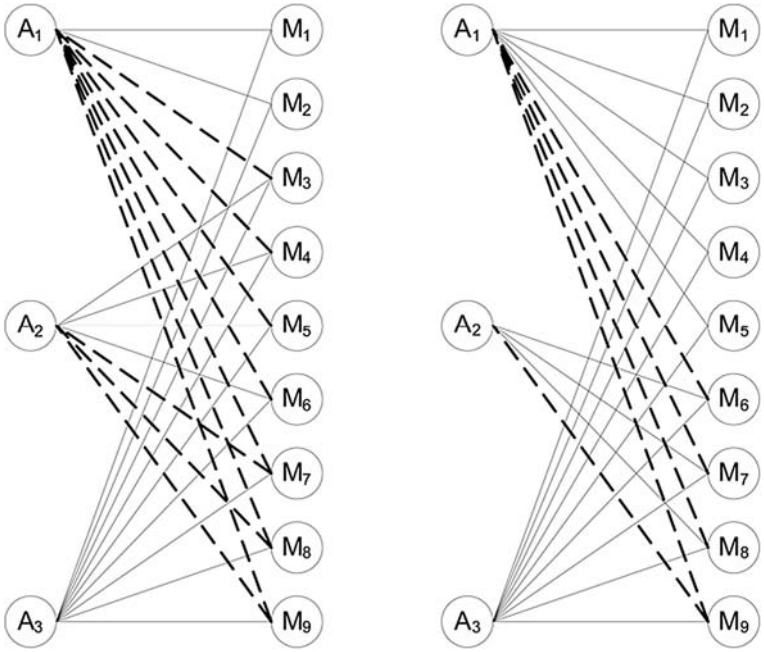

Fig. 2 Exemplary distribution of skills

right-hand side system. Thus, the operational flexibility is slightly higher for the right-hand side system.

For a comparative study of the efficiency of alternative approaches, we must also take an effort factor as a basis. In the process of collecting and passing on change requests, the effort mainly consists of communication, correction, and preparation activities of different duration that occur between the actors. We refer to these expenses collectively as transaction costs (Picot 1982; Hildenbrand et al. 2007). The dashed edges represent the transaction costs that arise between a requirement and an actor who is not primarily competent to carry out the respective modification. Eventually, the capable actors' labor expenses[1] are incurred. The modification costs, consisting of labor and transaction costs, are shown in **Fig. 3** with a general cost matrix for the operations M_1–M_9 as an example.

5 Method for the Flexible Modification of Business Processes

We consider a method to be a systematic approach for the achievement of predefined goals. A method supports the user with behavioral rules and instructions, which are based on certain principles. This understanding is based on the analysis of the works by Jayaratna (1994), Greiffenberg (1997, 2003), and Braun et al. (2004) and is not to be compared with a methodology (Hildenbrand et al. 2005; Ovaska 2005; Greiffenberg 1997; Becker et al. 2001) or a framework (Jayaratna 1994). For an in-depth definition and differentiation we refer to Greiffenberg (2003).

From a scientific perspective, the development of the method belongs to the method construction discipline (Greiffenberg 2003; Hevner et al. 2004). Following Gutzwiller (1994), the present work defines the approach (time schedule of activities and their dependencies) and actors (roles) as well as requirements for tool support. The existing meta-model of the method and a detailed explanation of artifacts and techniques only bear secondary relevance to the understanding of the conducted simulation. Therefore, we abstained from going into this further in view of the limitation of this contribution's size.

The presented method follows the principles of Web 2.0, enabling the simple modification of software systems (Schroth and Janner 2007). It draws upon the concepts of Model Driven Development (MDD). Frequently, the program code does not have to be changed manually. Simple changes can be made by persons without skills in software development. As a result of the implementation of changes being closely related to a specific department, the communication effort to mediate and the control functionality requirements can be reduced. More complex changes and the development of software services within a process still require advanced programming knowledge. Technological implementations of this philosophy can be found in Hoyer et al. (2009) and Kuropka et al. (2008), which however lack the formal evaluation in terms of flexibility and efficiency of the approaches.

5.1 Differentiation from the Component-Based Approach

The platform stands face to face with traditional component-based software development (component approach). The latter can be implemented for example with the Java Platform Enterprise Edition (Java EE) or .NET and C++ e.g. by using singletons. The differences between the component approach and the platform approach are (Roman 1999; Matena et al. 2001):

- Specifications for components are not directly derived from business process models, but usually from UML use cases. On the process platform, the implementation is done based directly on the business process activities.
- The business process is not directly represented in the software. The business logic is located in the components or by using a model view controller approach (MVC) in the controller. The business process is modeled and executed on the platform. The components are accessed by the user interface or, in case of the MVC approach, by the controller. On the platform, service operations are accessed by the business process.
- The processes are only implicitly implemented in the software so that a simple modification requirement may cause a complex modification of the software. Process changes can be implemented in the platform very easily.
- Change requests must be communicated via a number of persons in order to be ultimately implemented by skilled software developers in nearly every case. On the platform, a number of changes can be implemented by predefined roles.

[1] These can be determined with approaches using Functional Size Measurements (Abran et al. 2002).

Modifications M_i / Actors A_j	1	2	3	4	5	6	7	8	9
1	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{16}	K_{17}	K_{18}	K_{19}
2	K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{27}	K_{28}	K_{29}
3	K_{31}	K_{32}	K_{33}	K_{34}	K_{35}	K_{36}	K_{37}	K_{38}	K_{39}

Modifications M_i / Actors A_j	1	2	3	4	5	6	7	8	9
1	2	2	2	2	2	5	5	5	5
2	0	0	0	0	0	3	3	3	5
3	4	4	4	4	4	4	4	4	7
Labor costs	*2*	*2*	*2*	*2*	*2*	*3*	*3*	*3*	*7*
Transaction costs	*0*	*0*	*0*	*0*	*0*	*5*	*5*	*5*	*10*

Fig. 3 *Left*: General cost matrix for the implementation of changes M_i by the actors A_j. *Right*: Example values for the system in Fig. 2 right. Shaded cells correspond to the dashed edges and the values represent transaction costs measured in cost units CU. Non-shaded cells represent direct modification costs or no relation (costs = 0 CU)

5.2 Actor Model and Distribution of Skills

For our approach, four basic actors are distinguished: user, configurator, process administrator, and developer. The distribution of the actors' skills is illustrated in **Fig.** 4. To simplify matters, the allocation of the change operation is carried out via categories; some categories may also be adopted by several actors. Analogously, the graph-based representation of the distribution of skills is illustrated in **Fig.** 5.

- *User*: The user has simple changing competencies with regard to the business process – *Category A*. He is given greater decision-making authority than in traditional software development, the latter being characterized by little direct opportunities for the user to intervene in the process. At the same time, increased knowledge of a process-oriented language is required.
- *Configurator*: The configurator does not exist in classical approaches, currently his tasks are in the area of an administrator or developer. The configurator is trained for the adaptation and verification of (executable) business processes. He has all user skills for conducting changes and has further rights to modify a process – *Category A and B*. He can release changes requested by users. He has the final say regarding the bundling and communication of change requirements.
- *Process administrator*: The process administrator is responsible for the receipt of bundled changes and the implementation of changes in the process models. He is responsible for most of the changes and can theoretically inherit changes of categories A, B, and C. He is the controlling authority regarding the work of the configurators and, for further changes, of the developers.

Language construct (to which the modification is related) / Modification type	Activity	Flow	Gate	Event	User interface	Service	Role
Add	AA	AF	AG	AE	AUI	AS	AR
Delete	DA	DF	DG	DE	DUI	DS	DR
Move	MA	MF	MG	ME	MUI	MS	MR
Adjust	AdA	AdF	AdG	AdE	AdB	AdS	AdR
Create	*N/A*	*N/A*	*N/A*	*N/A*	CUI	CS	*N/A*

Category A Category B Category C Category D

Fig. 4 Distribution of skills for the adaptation of business processes, categorization for the process platform. Not all language constructs from Fig. 1 are relevant to our configuration

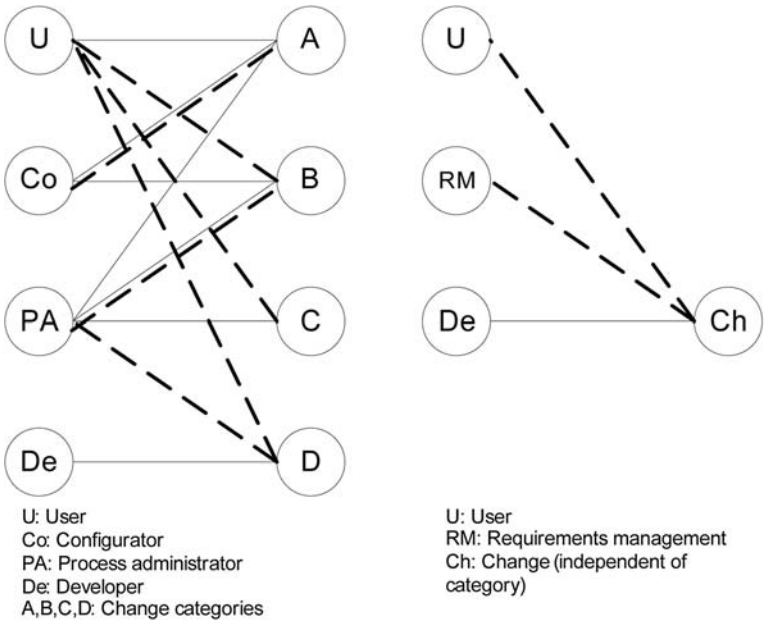

Fig. 5 Graph-based distribution of skills for the platform approach (*left*) and the component approach (*right*)

- *Developer*: The developer creates software services and their orchestration, supporting the business process – *Category D*. The tasks are similar to those of a traditional software developer. The focus, however, is mainly set

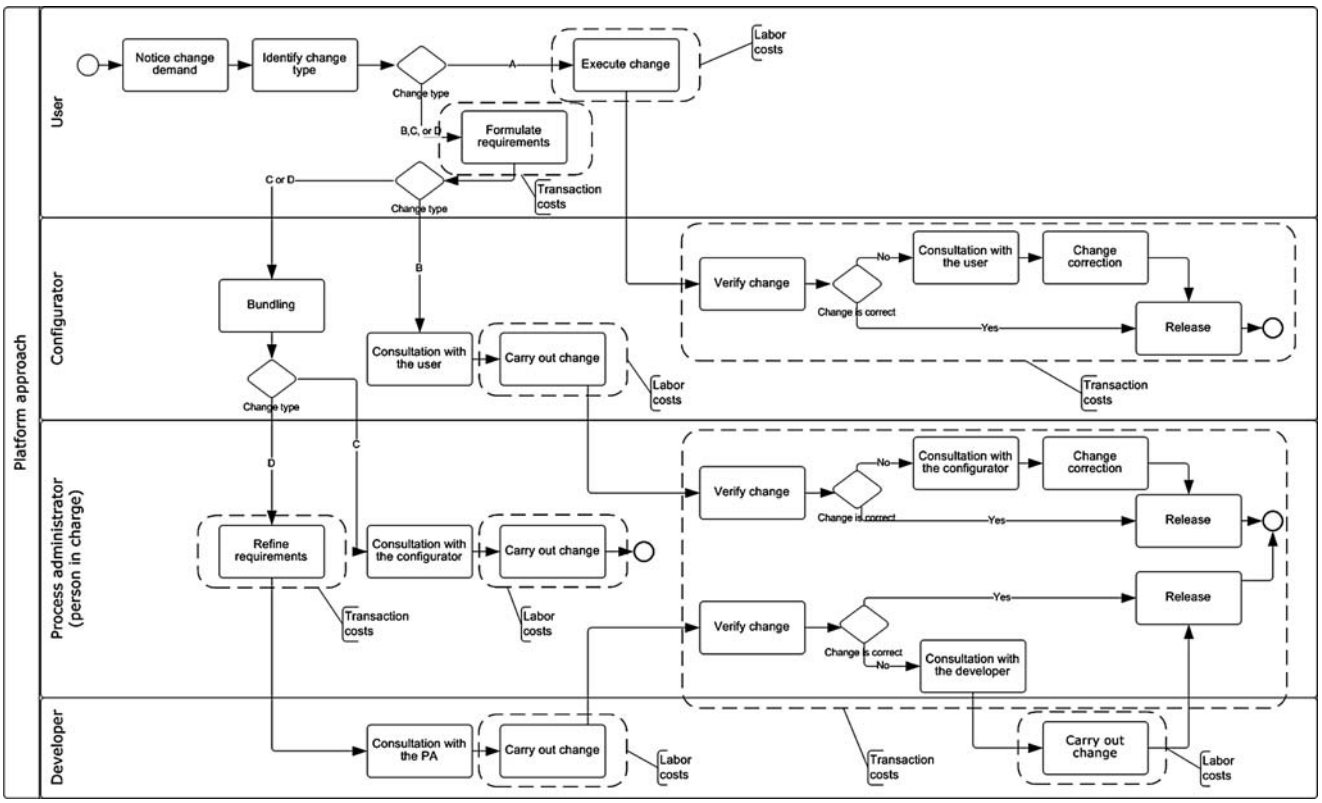

Fig. 6 Process model of the method for flexible modification of business processes. Labor and transaction costs are associated with the activities

on the adaptation and development of services.

5.3 Approach

The process model of the method allows for a structured treatment of the change requests based on the presented distribution of skills. The decision which actors have to carry out certain modifications in which manner is based on the required type of change. The process model is illustrated in **Fig. 6**.

5.4 Tool Support

Only the support of the method by a suitable tool allows for compliance with the presented approach and the division of tasks based on the defined roles. The presented approach is based on a platform implementation with various components to support the modeling and technical implementation of business processes (Elhadad et al. 2008; Schönherr et al. 2008). These include:

- *Portal*: Access to the process modeling through a web-based portal which offers different functions depending on the change options of the roles.
- *Modeling tool*: Based on an extended version of the Business Process Modeling Notation (BPMN 1.0), the users are able to take on the modeling and manipulation of processes. As part of the modeling tool existing models and services can be browsed.
- *Semantic repository*: In the repository, additional reference models are provided in addition to the customers' models (processes, data, organization), which can be adapted during modeling. Furthermore, a number of web services exist to support the processes.

For a better understanding, we present a short list of possible tool functionalities in the following:

1. *Syntactic modeling support*: Creating syntactically correct models can be difficult, especially for less trained users (e.g. the role *user*). In support of the users, the presented approach therefore offers situational advice and modeling proposals (e.g. potential successors to an activity, review of the number of edges for decision nodes, etc.; Born et al. 2007; Koschmider 2007; Soffer et al. 2008)
2. *Semantic service search*: For locating existing services, two methods are used. Existing data of the process are used to perform a match on the basis of semantically annotated service descriptions and to identify candidates for the support of the activity/activities. In addition, the current process can be analyzed with methods from NLP (Natural Language Processing) and the concepts can be compared with a terminology of the customer (if available). In order to overcome the heterogeneity of different terminologies, methods from the so-called ontology alignment are used. (Elhadad et al. 2008; Kuropka et al. 2008; Rake et al. 2009)
3. *Process model adaptation*: The definition, to what extent the modeler is allowed to make adjustments to the workflow model, is included in the process model adaptation. Approaches for the specification of these operational steps of process model adaptation can be found in adaption patterns for workflows (Weber et al. 2007), in specialization rules for the derivation of process models (Soffer et al. 2007), and in approaches on the extension of process models with ontology-based annotations to

Platform approach

Modifications M_j / Actors A_i	Category A	Category B	Category C	Category D
User	2	2	2	2
Configurator	2	3	0	0
Process Administrator	0	2	5	3
Developer	0	0	0	8
Labor costs	*2*	*3*	*5*	*8*
Transaction costs	*2*	*4*	*2*	*5*

Component approach

Modifications M_j / Actors A_i	Category A,B,C,D
User	2
Change Management	1
Developer	5
Labor costs	*5*
Transaction costs	*3*

Fig. 7 Cost matrices for the alternative approaches to software modification (cost in CU)

express constraints and rules for the change process (Soffer et al. 2008).

6 Case-Study-Based Simulation and Results

In order to illustrate the simulation, in the following section we will explain the design, the case study, and the change scenarios as well as outline and discuss the results.

6.1 Simulation Design

To evaluate the platform-based method, we conduct a simulation (Kellner et al. 1999; Andres and Zmud 2001; Kleijnen et al. 2005). As the basis of the simulation we consider change scenarios, which have been taken from real change requirements of business processes. These scenarios are explained below. The options available for adaptation are defined by the modification operations.

The question is how easily the original process can be adapted with regard to the above mentioned change scenarios: once following a current software development model (component approach) and once following our method. The effort required for implementing a change consists of the specific expenses of the executing actors and the transaction costs which arise from the communication between actors during the transmission of change requests (see Sect. 4).

We assume different labor costs for the platform approach depending on the category of the change. In the component approach, every change will be implemented by a developer; therefore, no differences exist between the various change categories.

Transaction costs depend on the category since different actors are involved in the changes of a category (see also **Fig. 5**). The corresponding cost matrices are illustrated in **Fig. 7** (transaction costs of the actors involved are shaded).

The weightings have been developed based on expert estimates since we are not aware of any empirical work showing this granularity of observation. We assume conservative values for the platform approach as it is still technologically maturing.

6.2 Case Study

In the context of current customer projects in the public sector, we could draw upon a basis of management processes. We identified the registration process for childcare facilities (which takes place in all municipalities nationwide) as a potential domain for the application of our approach. This process and its context are suitable for the analysis of our approach since on an abstract level the individual processes in local communities are the same: parents register their children for various care facilities, an administrative unit determines the legitimacy and possibly special needs of the children and assigns day care places, and care facilities finally accept and take on the children. At the implementation level, however, the process can be realized differently. The provision of this registration process as a service by a single IT service provider may be an attractive model due to the above-mentioned advantages – for the individual municipalities as well as for the IT service provider. In order to enable the successful consumption of the process by many municipalities it is important for the offering company to be able to quickly and easily adjust the basic process to the individual differences between the municipalities. To verify the presented approach, we selected typical change scenarios of varying complexity from the domain. These are carried out in a simulation in order to quantitatively assess flexibility properties. In a comparison with an existing software development model (component approach) we ultimately highlight the benefits and risks of our approach.

6.3 Change Scenarios

We consider three change scenarios that may occur in practice according to our analysis of various municipalities.

- *Change Scenario 1: Early end of registration deadline*[2] – The expiry date of a central registration will be moved to an earlier date. This requires a change of the corresponding temporal event in the workflow. The process flow is then resumed earlier accordingly.
- *Change Scenario 2: Central place assignment instead of individual prioritization*[3] – The parents of the children should not be allowed to prioritize child care services in the online reservation on their own anymore. Instead, the administrative unit carries out the prioritization and assignment of childcare offers (central planning of the childcare). Parents only register their children online; preferential facilities can no longer be specified.
- *Change Scenario 3: Additional data transmission for statistical analysis*[4] – Presently, the transmission of personal data (of the children) by the childcare facility to the administrative unit can also be performed manually (e.g.

[2] One operation overall; modification operation involved: AdE.
[3] 40 operations overall; modification operations involved: MA, AdA, MF, AE, AF, MG, ME, DA, DF, DE, AdF, AdE.
[4] 12 operations overall; modification operations involved: AS, AdA, AdS, MF, AF, AR, AE, AA.

Table 2 Results of the simulation: Modification costs of the alternative approaches in three change scenarios (costs in CU) and their individual operational flexibility

Scenario	Category A	Category B	Category C	Category D	Total
			Scenario 1		
Number of changes (total)	0	1	0	0	1
LC Platform	0	3	0	0	3
TC Platform	0	4	0	0	4
LC Components	–	–	–	–	5
TC Components	–	–	–	–	3
			Scenario 2		
Number of changes (total)	3	28	9	0	40
LC Platform	6	84	45	0	135
TC Platform	6	112	18	0	136
LC Components	–	–	–	–	200
TC Components	–	–	–	–	120
			Scenario 3		
Number of changes (total)	4	1	6	1	12
LC Platform	8	3	30	8	49
TC Platform	8	4	12	5	29
LC Components	–	–	–	–	60
TC Components	–	–	–	–	36
Operational flexibility					
Platform approach	1.53				
Component approach	1				

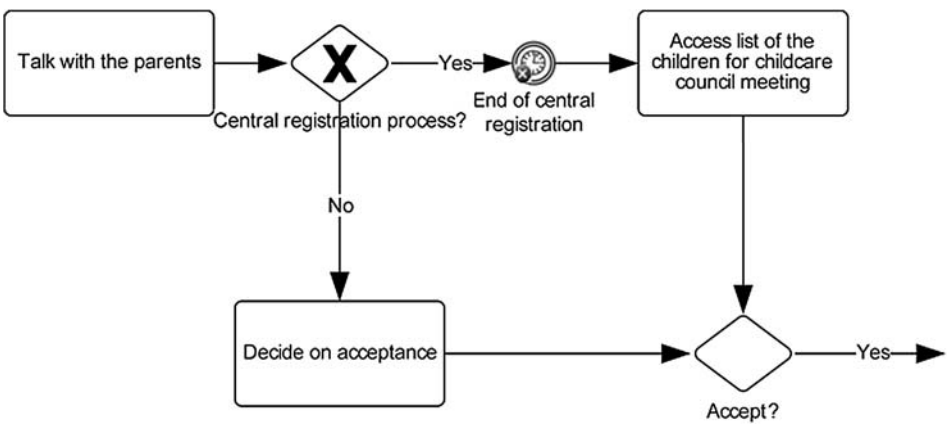

Fig. 8 Detail of change scenario 1

by mail) – in future, the transmission will be done electronically. This makes it necessary to provide corresponding services and to incorporate them into the process. In addition to the transmission of personal data to the administrative unit or the Department of Children and Families respectively, the data should be transmitted to the local government in order to create reports on this basis for enabling statistical analysis. This requires not only new activities and control flows, but also adding a new role (local government) in the process.

6.4 Implementation of the Simulation and Summary of Results

The simulation was carried out for the two alternative approaches based on the change scenarios described in Sect. 6.3; i.e. we carried out a total of six simulation runs (three runs for each approach).

The results of the simulation are summarized in **Table 2**. Due to the limitation of this contribution's size we only briefly discuss each scenario.

In change scenario 1, only change operation AdE had to be performed to set the timer in the time event correspondingly (see **Fig. 8**). In the component approach the developer is involved for a relatively small change, causing additional expenses in the LC compared to the platform approach. However, the more complex actor structure leads to slightly higher coordination costs in the platform approach.

Change scenario 2 (see **Fig. 9**) includes the movement of the process elements which describe the prioritization of childcare offers since these activities are no longer carried out by the role of the parents, but by the administration. The LC for the platform approach amount to 135, which is well below the component approach (= 200). This is due

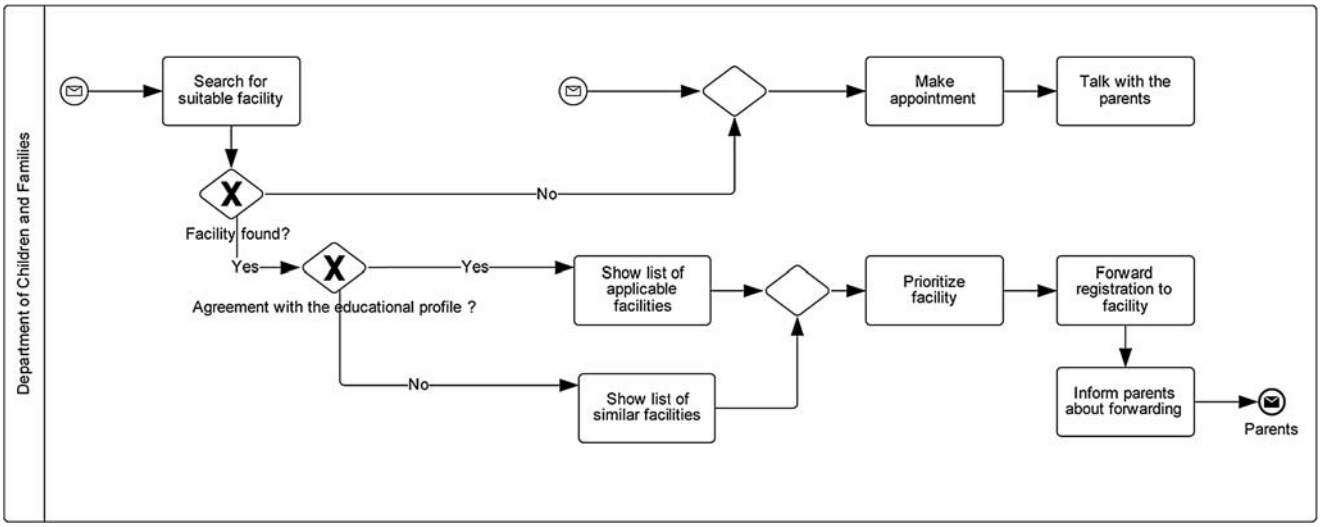

Fig. 9 Detail of change scenario 2

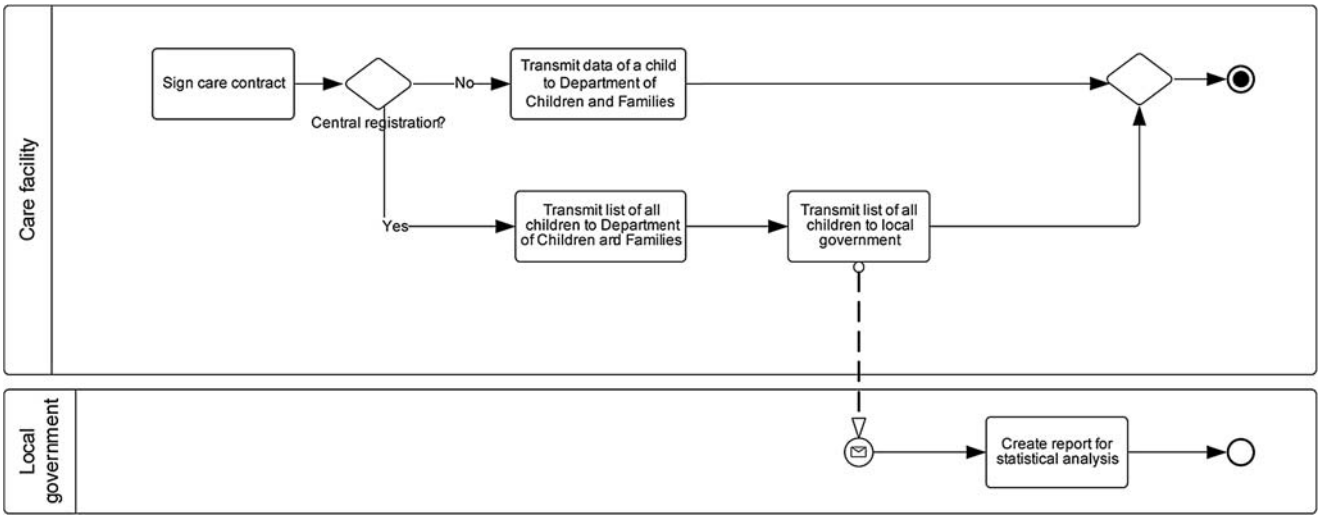

Fig. 10 Detail of change scenario 3

to the fact that many change operations, such as moving, adding, adapting, etc. are performed by the configurator without generating work for process administrators or developers. The platform approach can thus avoid "small" changes for the developers. Again, the higher TC are a result of the more complex actor model.

For change scenario 3 (see **Fig. 10**) we can see that the platform approach leads to the fact that the LC for the overall change reduced by about 20% compared to the component approach ($49 < 60$). Although in this scenario changes necessarily require the participation of developers for both approaches, the expenses are not higher in the platform approach in terms of the LC compared to the component approach. However, the development effort for the adaptation of a service is higher in the platform approach (category D). As to the other change operations, the platform approach provides the advantage that many change requests do not even reach the developer because they could already be implemented by the user or the configurator. This will also result in slightly lower TC in the platform approach.

Operational flexibility can be determined for each approach according to the definition given in Sect. 3. In the component approach only the developer can implement a change. Thus, for this approach we obtain an operational flexibility of $30/30 = 1$. Due to subsumption of the categories A to B and B to C, we receive a different picture for the platform approach. Here, the operational flexibility is $46/30 = 1.53$ and with 53% is much higher than for the component approach.

All in all, the platform approach was able to reach a better result in the LC for all three simulated change scenarios. However, the platform approach causes higher TC in two of three change scenarios. The increased operational flexibility in the platform approach can be explained by the increased distribution of modification operations to various actors. Employees without further programming knowledge can now make (simple) modifications on their own This quality becomes especially apparent in case of capacity shortages for specific actors, such as developers. Regarding the labor costs a cost reduction can be shown for the scenarios in the platform

Zusammenfassung / Abstract

Oliver Holschke, Jannis Rake, Philipp Offermann, Udo Bub

Improving Software Flexibility for Business Process Changes

In times of continuous change, companies need to adjust their business processes to gain sustainable competitive advantage. Resulting changes in the company's IT currently require the involvement of developers from departments that are mostly not aligned with the business. These changes often result in high transaction and labor costs. The article presents a platform-based method to adjust business processes with the aim of increasing both efficiency and flexibility compared to current approaches. The core of our work is an evaluation against traditional component-based software development using a sound simulation model. Three real-world scenarios of business process change show that – despite a slight increase in transaction costs – our suggested method decreases labor costs while increasing operational flexibility.

Keywords: Business processes, Flexibility, Method, Process platform, Service-oriented architecture, Software-as-a-Service

approach – this, however, is usually traded off for an increase in transaction costs as a result of the increased interactions between multiple actors.

6.5 Limitations of the Model and Results

The graph-based model for transactions may not sufficiently reflect the complexity of software modifications. The TC are based only on simple relations in the graph. This assumption is legitimate only if the described change operations remain limited to certain areas in the organization (e.g. that changes are only communicated between user and configurator) and do not cause interdependencies beyond these regions. The relaxation of this restriction would increase the TC again and possibly neutralize any advantage gained.

The incurred costs were assumed on the basis of expert evaluations since an empirical basis is not available in the required granularity. Further analyses based on the presented model can be carried out by means of future empirical surveys.

The platform approach will possibly not be able to realize radical changes efficiently, while the component approach may be better suited. Under the assumption that radical changes occur with limited frequency, the proposed approach can be sustainably efficient. The frequency assumption may be inaccurate. Due to the specific flexibility of our approach, which assumes a basic process and allows for certain change operations, deviant extents of change requirements have to be assessed accurately. If changes exceed a certain quality, especially those requiring the costly adjustment of the code, then a change request by the configurator may be restricted or can even be rejected by the process administrator.

The realization of the platform's service-orientation also carries additional costs in the use of process engines and service management as well as a non-negligible communication overhead in the form of XML data transmission. Alternative solution architectures need to be evaluated.

7 Conclusion and Outlook

Process platforms (in terms of PaaS solutions) founded on diagram-based and user-centered interfaces and composable services can be seen as an opportunity to further improve the changeability of business processes. In this article, we evaluated a platform based method which follows these principles. For the evaluation a model consisting of actors and types of modification was developed, which also allows for the formulation of labor and transaction costs. By means of this model we carried out an analysis of software change approaches with regard to flexibility and efficiency.

The simulation of three real-world change scenarios showed that the platform approach has advantages in terms of flexibility and efficiency compared to a traditional component-oriented approach for software change. The option that changes can be conducted by actors who are more closely involved in the business side, can help reduce the above described business-IT gap.

To demonstrate the general robustness of the approach it is necessary to conduct further empirical research. Potential change scenarios should be modeled as stochastic processes which allow for the analysis of the performance of alternative software modification approaches. In supplementing research the approaches which enable high operational flexibility and efficient implementation for certain assumed distributions of change requirements can be identified.

References

Abran A, Silva I, Primera L (2002) Field studies using functional size measurement in building estimation models for software maintenance. Journal of Software Maintenance: Research and Practice 14(1):31–64

Aier S, Winter R (2009) Virtual decoupling for IT/business alignment – conceptual foundations, architecture design and implementation example. Business & Information Systems Engineering 1(2):175–191

Allweyer T (1998) Adaptive Geschäftsprozesse: Rahmenkonzept und Informationssysteme. Gabler, Wiesbaden

Alter S (2008) Defining information systems as work systems: implications for the IS field. European Journal of Information Systems 17:448–469

Andres HP, Zmud RW (2001) A contingency approach to software project coordination. Journal of Management Information Systems 18(3):41–70

Baldwin CY, Clark KB (2000) Design rules: the power of modularity. MIT Press, Cambridge

Becker J, Holten R, Knackstedt R, Hansmann H, Neumann S (2001) Konstruktion von Methodiken – Vorschläge für eine begriffliche Grundlegung und domänenspezifische Anwendungsbeispiele. Arbeitspapier

Berry WL, Cooper MC (1999) Manufacturing flexibility: methods for measuring the impact of product variety on performance in process industries. Journal of Operations Management 17(2):163–178

Born M, Dorr F, Weber I (2007) User-friendly semantic annotation in business process modeling. Lecture Notes in Computer Science 4832:260

Braun C, Hafner M, Wortmann F (2004) Methodenkonstruktion als wissenschaftlicher Erkenntnisansatz. Universität St. Gallen, Institut für Wirtschaftsinformatik

Brehm L, Heinzl A, Markus ML (2001) Tailoring ERP systems: a spectrum of choices and their implications. In: Proceedings of 34th Hawaii international conference on system sciences (HICSS'01), Hawaii, IEEE

Buxmann P, Hess T, Lehmann S (2008) Software as a service. Wirtschaftsinformatik 50(6):500–503

Chan YE, Reich BH (2007) IT alignment: what have we learned? Journal of Information Technology 22(4):297 -315

Clemons EK (1986) Information systems for sustainable competitive advantage. Information and Management 11(3):131–136

Cyert RM, March JG (1963) A behavioral theory of the firm, 1st edn. Prentice-Hall, New Jersey

Davenport TH, Short JE (1990) The new industrial engineering: information technology and business process redesign. Sloan Management Review 31(4):11–27

Dreyfus D, Iyer B (2008) Managing architectural emergence: a conceptual model and simulation. Decision Support Systems 46:115–127

Elhadad M, Balaban M, Sturm A (2008) Effective business process outsourcing: the Prosero approach. International Journal of Interoperability in Business Information Systems 3(1)

Ferstl OK, Sinz EJ (1998) Grundlagen der Wirtschaftsinformatik, 3rd edn. Oldenbourg, Munich

Gebauer J, Lee F (2008) Enterprise system flexibility and implementation strategies: aligning theory with evidence from a case study. Information Systems Management 25(1):71–82

Gersick C (1991) Revolutionary change theories: a multilevel exploration of the punctuated equilibrium paradigm. Academy of Management Review 16(1):10–36

Gottschalk F, Aalst WMP, Jansen-Vullers MH, La Rosa M (2008) Configurable workflows. International Journal of Cooperative Information Systems 17(2):177–221

Greiffenberg S (1997) Methodenentwicklung in Wirtschaft und Verwaltung. Dr. Kovač, Hamburg

Greiffenberg S (2003) Methoden als Theorien der Wirtschaftsinformatik. In: Uhr W, Esswein W, Schoop E (eds) Wirtschaftsinformatik Band 2 pp 947–968

Gutzwiller TA (1994) Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen. Physica, Heidelberg

Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Quarterly 28(1):75–105

Hildenbrand T, Behm A, Rashid A, Geisser M (2005) Entwicklungsmethodiken zur kollaborativen Softwareerstellung. Universität Mannheim, Mannheim

Hildenbrand T, Rothlauf F, Heinzl A (2007) Ansatze zur kollaborativen Softwareerstellung. Wirtschaftsinformatik 49(0):72–80

Hong K-K, Kim Y-G (2002) The critical success factors for ERP implementation: an organizational fit perspective. Information & Management 40:25–40

Hoyer V, Janner T, Schroth C, Delchev I, Urmetzer F (2009) FAST platform: a concept for user-centric, enterprise class mashup. In: Proceedings of 5th conference of professional knowledge management: gesellschaft für Informatik (GI), Solothurn

Hoyer V, Stanoesvka-Slabeva K, Janner T, Schroth C (2008) Enterprise mashups: design principles towards the long tail of user needs. In: Proceedings of IEEE international conference on services computing (SCC 2008), Honolulu, IEEE Computer Society

Iravani SM, Oyen MPV, Sims KT (2005) Structural flexibility: a new perspective on the design of manufacturing and service operations. Management Science 51(2):151–166

Jayaratna N (1994) Understanding and evaluating methodologies: NIMSAD, a systematic framework. McGraw-Hill, London

Kellner MI, Madachy RJ, Raffo DM (1999) Software process simulation modeling: why? what? how? Journal of Systems and Software 46(2/3):91–105

King WR, Grover V, Hufnagel EH (1989) Using information and information technology for sustainable competitive advantage: some empirical evidence. Information & Management 17:87–93

Kleijnen JPC, Sanchez SM, Lucas TW, Cioppa TM (2005) A user's guide to the brave new world of designing simulation experiments. INFORMS Journal on Computing 17(3):263–289

Koschmider A (2007) Ähnlichkeitsbasierte Modellierungsunterstützung für Geschäftsprozesse. Universitaetsverlag Karlsruhe, Karlsruhe

Kumar RL (2004) A framework for assessing the business value of information technology infrastructures. Journal of Management Information Systems 21(2):11–32

Kuropka D, Tröger P, Staab S et al (2008) Semantic service provisioning. Springer, Heidelberg

Lyytinen K (1987) A taxonomic perspective of information systems development: theoretical constructs and recommendations. In: Boland R, Hirschheim R (eds) Critical issues in information systems. Wiley, Chichester, pp 3–43

Malone TW, Crowston K (1994) The interdisciplinary study of coordination. ACM Computing Surveys 26:1

Masak D (2006) IT-alignment. Springer, Heidelberg

Matena V, Krishnan S, Michiel LD (2001) Applying enterprise JavaBeans: component-based development for the J2EE platform. Addison-Wesley, Amsterdam,

Mockus A, Votta LG (2000) Identifying reasons for software changes using historic databases. In: Proceedings of proceedings of the international conference on software maintenance (ICSM'00), IEEE Comput Soc

Moitra D, Ganesh J (2005) Web services and flexible business processes: towards the adaptive enterprise. Information and Management 42(7):921–933

Ovaska P (2005) Working with methods. In: Vasilecas O (ed) Information systems development: advances in theory, practice and education. Springer, Heidelberg

Picot A (1982) Transaktionskostenansatz in der Organisationstheorie: Stand der Diskussion und Aussagewert. Die Betriebswirtschaft 42:267–284

Rake J, Holschke O, Levina O (2009) Enhancing semantic service discovery in heterogeneous environments. In: Proceedings of business information systems (BIS), Poznan. Springer, Heidelberg

Regev G, Bider I, Wegmann A (2007) Defining business process flexibility with the help of invariants. Software Process Improvement and Practice 12:65–79

Remme M (1997) Konstruktion von Geschäftsprozessen: Ein modellgestützter Ansatz durch Montage generischer Prozesspartikel. Gabler, Wiesbaden

Ried S, Garbani J-P, Bartels A, Lissermann M (2009) Platform-as-a-service market sizing. Forrester Research

Roman E (1999) Mastering enterprise Javabeans: and the Java 2 platform, enterprise edition. Wiley, New York

Rosemann M, Aalst WMP (2007) A configurable reference modelling language. Inf. Syst. 32(1):1–23

Schönherr M, Holschke O, Offermann P, Bub U (2008) Werkzeuggestützte Referenzmodellanwendung im SOA-Entwurfsprozess. Informatik 2008. Beherrschbare Systeme – dank Informatik. In: Proceedings, vol 1, Munich. GITO Verlag, Berlin

Schroth C, Janner T (2007) Web 2 0 and SOA: converging concepts enabling the internet of services. IT Professional 9:36–41

See Pui Ng C, Gable GG, Chan T (2002) An ERP-client benefit-oriented maintenance taxonomy. Journal of Systems and Software 64(2):87–109

Sethi AK, Sethi SP (1990) Flexibility in manufacturing: a survey. The International Journal of Flexible Manufacturing Systems 2:289–328

Soffer P, Kaner M, Wand Y (2008) Assigning ontology-based semantics to process models: the case of petri nets. In: Proceedings of CAiSE 2008, Montpellier. Springer, Heidelberg

Soffer P, Reinhartz-Berger I, Sturm A (2007) Facilitating reuse by specialization of reference models for business process design. In: 8th workshop on business process modeling, development, and support (BPMDS'07) in conjunction with the 19th international conference on advanced information systems engineering (CAiSE'07)

Swanson EB (1976) The dimensions of maintenance. In: Proceedings of the 2nd international conference on software engineering, San Francisco. IEEE Comput Soc, Los Alamitos

Turkay E, Gokhale AS, Natarajan B (2004) Addressing the middleware configuration challenges using model-based techniques. In: Proceedings of southeast regional conference. ACM, New York

Weber B, Rinderle S, Reichert M (2007) Change patterns and change support features in process-aware information systems. In: Proceedings of CAiSE 2007. Springer, Heidelberg