Softw Syst Model (2015) 14:1159–1188
DOI 10.1007/s10270-014-0412-6

SPECIAL SECTION PAPER

Design science in action: developing a modeling technique for eliciting requirements on business process management (BPM) tools

Ilia Bider · Erik Perjons

Received: 7 October 2012 / Revised: 1 April 2014 / Accepted: 8 April 2014 / Published online: 9 May 2014

Abstract Selecting a suitable business process management (BPM) tool to build a business process support system for a particular business process is difficult. There are a number of BPM tools on the market that are available as systems to install locally and as services in the cloud. These tools are based on different BPM paradigms (e.g., workflow or case management) and provide different capabilities (e.g., enforcement of the control flow, shared spaces, or a collaborative environment). This makes it difficult for an organization to select a tool that would fit the business processes at hand. The paper suggests a solution for this problem. The core of the solution is a modeling technique for business processes for eliciting their requirements for a suitable BPM tool. It produces a high-level, business process model, called a "step-relationship" model that depicts the essential characteristics of a process in a paradigm-independent way. The solution presented in this paper has been developed based on the paradigm of design science research, and the paper discusses the research project from the design science perspective. The solution has been applied in two case studies in order to demonstrate its feasibility.

Keywords Business process modeling · Workflow · Case management · Shared space · Design science

Communicated by Dr. Selmin Nurcan.

I. Bider (✉) · E. Perjons
DSV, Stockholm University, Forum 100,
16440 Kista, Stockholm, Sweden
e-mail: ilia@dsv.su.se

E. Perjons
e-mail: perjons@dsv.su.se

I. Bider
IbisSoft AB, Box 19567, 10432 Stockholm, Sweden

1 Introduction

1.1 The problem to be addressed

A business process support (BPS) system is a system that helps process participants to run their processes efficiently. To build such a system from scratch is time-consuming and requires a great deal of software engineering. The latter can be greatly diminished by employing a business process management (BPM) tool that includes high-level means for building BPS systems. However, selecting a suitable BPM tool for building a BPS system for a particular process is not a trivial task. The major difficulty lies in the fact that there are numerous BPM tools on the market, and they implement different paradigms upon which one can build a BPS system.

The mainstream paradigm in this domain is workflow-oriented thinking [44]. Besides the workflow paradigm, there are some less-widely known paradigms, such as case management [43], including adaptive case management [42]. Usually, providers of BPM tools build their tools based on one of the paradigms—some of the tools are workflow-based, while others are based on case management.

The suitability of a particular paradigm for creating computerized support depends upon the business process in question. Some business processes can be streamlined and optimized, making workflow-based tools a perfect match for building BPS systems. For others, a social software tool, such as a wiki, can be an appropriate choice. Therefore, selecting a BPM tool to build a BPS system for a particular process requires an understanding of the nature of the process at hand. For example, employing a workflow-based tool requires the splitting of the process into a number of predefined operations/activities that can be ordered in a systematic way, whereas the use of a wiki-type service does not even require the identification of the operations that are to be carried out. The latter alternative could be more suitable for emergent or loosely structured processes, for which it is hard to predefine the operations/activities and the order in which they should be carried out.

Trying several paradigms to discover which one is the most suitable for the given process can be costly and impractical. The problem is to find/develop a method that will allow the business to elicit the requirements on the capabilities to be provided by the BPM tool without going into the details of the process, for example, without building a detailed model of this process. Besides being too costly, building a detailed model requires the choice of a language/notation, for example, BPMN. As most languages are tightly coupled with a specific paradigm (e.g., BPMN is coupled with workflow), choosing a specific language may later influence the decision on which type of BPM tools will be selected for implementing computerized support.

The need for finding practical methods for selecting appropriate BPM tools becomes more urgent with growing number of BPM tools offered as cloud services, see for example, [1,7,37,39]. The shift to cloud computing in the BPM area means increased availability of inexpensive BPM tools, delivered as services that are ready for the deployment at a moment's notice. This makes the use of BPS systems affordable, even for small companies and organizations. In addition, it allows business people to employ a BPS system as a service at the department level. The burden of having to consult with the IT department about the existence of an appropriate IT infrastructure to support a particular tool they want is lifted, because the only infrastructure they need is a Web browser. The same applies to financial concerns; it is possible to test a new service without high initial investments. The decision about which BPM service to choose for some processes can be decided at the department level, without the involvement of IT people, who may not be present or may not be sufficiently competent in small enterprises and organizations.

1.2 Existing solutions

We have not found a practical method for dealing with the problem outlined in Sect. 1.1 in the literature. The works that deal with the issues of choosing BPM tools only deal with choosing a tool from the same paradigm, e.g., workflow [44], and/or only deal with selecting a tool using technical and not business criteria. However, the literature does identify the limitations in the scope of the applicability of particular paradigms for building computerized support, for example, see definition of "workflowability" in Baresi et al. [8]. This type of works can be useful when developing a solution for the problem at hand.

1.3 Suggested solution

In this paper, we propose the use of a high-level model of the process at hand for determining the capabilities of a suitable BPM tool for this process. The level of details in the model should be sufficient for gaining an understanding of the kind of tools that would be appropriate for this process, but no more. We believe that additional details should be investigated when the choice of tool has been made, using the appropriate means for that tool. Although the model we propose is a high-level model, it should not be focused only on the flow of operations/activities, but should also include several additional perspectives, such as information flow, and the needs of communication/collaboration in the process.

For creating this model, we have used a special modeling technique called "step-relationship" modeling. As the name points out, the model is defined in terms of steps and the relationships between them. Steps represent work packages or relatively large chunks of work, while relationships represent different kind of relations between the steps in the process, such as the input/output relationship, the possibility of parallel execution, and the presence of intersecting teams. The presence or absence of particular kinds of relationships is then used for establishing the requirements on a BPM tool to be used for building a BPS system.

Although we outline the entire approach, from building a model to choosing a BPM tool based on the analysis of the model, the focus of this paper is on describing the step-relationship modeling technique and on explaining the use of it in practice. Other components of our approach need to be developed and evaluated further.

The research presented in this paper is based on our experience of building BPS systems [5,10,11,14], as well as on our experiences in the sales and marketing of our own BPM tool, called *iPB* [28].[1]

1.4 Research methodology and structure of the paper

The research reported in this paper is being conducted using DSR principles [27,36]. More precisely, we use the DSR interpretation given in [12,13] that considers a DSR project as movement in and between two worlds: (a) the real world of specific problems and solutions in local practices, and (b) the abstract world of generic problems and solutions.

The paper is written according to the following plan. In Sect. 2, we introduce the main concepts related to business processes and BPS systems. In Sect. 3, we give our interpretation of DSR. In Sect. 4, we describe the problem and outline the solution, based on the DSR principles from Sect. 3.

[1] Note that the step-relationship model suggested in this paper has no direct connection to the models used in our BPS systems, and the modeling techniques used in *iPB*.

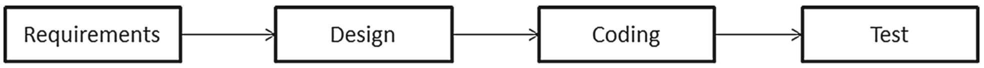

Fig. 1 A plan/template for handling a situation when there is a need to develop a customized software system

Sections 5 to7 present a detailed description of the components of the solution. Section 5 describes elements of the step-relationship process model. Section 6 lists and describes the capabilities that can be provided by a BPM tool. Section 7 presents a procedure for establishing which capabilities are required for a particular process through analysis of its step-relationship model. Section 8 discusses cases in which our solution has been tested, and lessons learned from them. Section 9 summarizes what has already been achieved, lists the challenges to be overcome, and draws plans for the future. Appendices provide additional details of the solution and test cases.

This paper is an extended version of our idea paper presented at the BPMDS 2012 conference [16]. In this version, Sects. 3 (Research Method-Design Science Research), 4 (From Identifying the Problem to Outlining the Solution), 8 (Demonstration and Evaluation) and “Appendixes 10 and 11” are completely new, while other sections have been revised and extended based on the feedback from the presentation.

2 Basic concepts and assumptions

This section introduces the basic concepts related to the business processes in sufficient detail that even a reader that is not familiar with the domain of BPM (Business Process Management) can follow the reasoning when we describe the problem and the suggested solution.

2.1 Business processes and process support systems

There are many definitions of the notion of business process, each of them highlighting different aspects, as described by [15]. The most common view of business processes is an operational one, in which the process is considered as a partially ordered sequence of activities aimed at reaching some goal [25,45]. This view serves as a basis for workflow-based BPS systems [44]. However, this view is not suitable for other types of BPS systems, for example, case-handling systems [43], and especially not for those that belong to the new paradigm of adaptive case management—ACM [42]. As our investigation is not limited to any particular type of BPS system, in this section, we introduce the basic concepts of business processes and BPS systems in a generic way that is compatible with different approaches to building BPS systems.[2]

The term *business process* encompasses two distinct concepts: *business process type* and *business process instance* (or case), here defined as follows:

– *business process type* is a plan/template for handling business situations of a certain type;
– *business process instance/case* is a situation being handled according to a plan/template suggested by a given business process type.[3]

A business process type (plan/template) can include information on any combination of the following:

– a situation that warrants application of the plan, i.e., that triggers the creation of a new instance;
– a goal to reach;
– sub-goals and an order in which they could/should be achieved (goal decomposition);
– operations/actions/activities that should be completed for achieving goals/sub-goals and the order in which they should be completed (operational decomposition);
– rules of responsibility/participation (both for sub-goals and operations);
– rules of collaboration/communication between participants pursuing common goals/sub-goals (communication/collaboration channels).

For example, consider the situation of the development of a customized software system for a particular customer. A general plan for handling this situation can be presented as a simplified flowchart, as shown in Fig. 1. To this flowchart, any number of details can be added, e.g., the first step in Fig. 1 should be carried out by requirements engineers, the second step should produce use case diagrams, and the third step should use Java as a programming language. The more details that are added, the more rigid the process will be. For example, setting the requirement that all programming should be done in Java will force the developers to use this language, even in cases where it does not fit, e.g., for the development of operating systems.

[2] The basis for the definitions in this section is a systems-oriented view of business processes presented in [10].

[3] The dichotomy type/instance is accepted in all dialects of the BPM literature. However, different terms can be used to express this dichotomy. For example “process” itself can be used to refer to the type, while case or run is used to denote the instance. Another term that is used to denote the type is process model, which is typical for an operational view. We have chosen to use the terms “type” and “instance” as they suit both our pragmatic and theoretical definition better.

The business process type (template) can reside in any combination of the following:[4]

– in the heads of staff members who participate in instances of this business process type (i.e., tacit knowledge). This knowledge guides the participants in the process in what is permitted, obliged, and/or prohibited, without requiring them to reflect about it;
– as written documents, including process maps and other kinds of process descriptions (i.e., explicit knowledge) stored on paper or electronically, e.g., in the form of Web-based hypertext. These documents contain explicit instructions of what is permitted, obliged, and/or prohibited;
– in software systems/services used to support the running of the instances of the process (i.e., built-in or embedded knowledge). The usage of such systems forces process participants to carry out some actions in a certain way and/or in a certain order.

In other words, the knowledge about processes can range from being completely tacit (e.g., residing in the heads of the process participants), to being totally explicit (e.g., being depicted in detailed process maps).

We define a business process support (BPS) system as a computer system that helps the participants of an instance of a business process to follow the plan/template defined by the business process type. It can, for example, automate certain operations or support coordination/collaboration between the workers who participate in the same instance of the process. Note that using a BPS system for supporting a process does not imply that the whole definition of the process needs to be built into this system, nor that the system needs to supports all operations included in the process. The amount and type of work the system should support depending on the nature of the process and its context.

2.2 Process structure

In this section, we define a number of concepts that describe the internal structure of business processes. These concepts are based on the following two assumptions:

1. Each process instance has a *goal* to reach, for example, sell to a customer one or more particular products from the company's assortment for a given price;
2. The goal of an instance can usually be decomposed into a number of *sub-goals* that could be pursued sequentially or in parallel.

Based on the above assumptions, we define additional concepts that can be used to describe the internal structure of a process in a manner that is independent of a particular view of business processes:

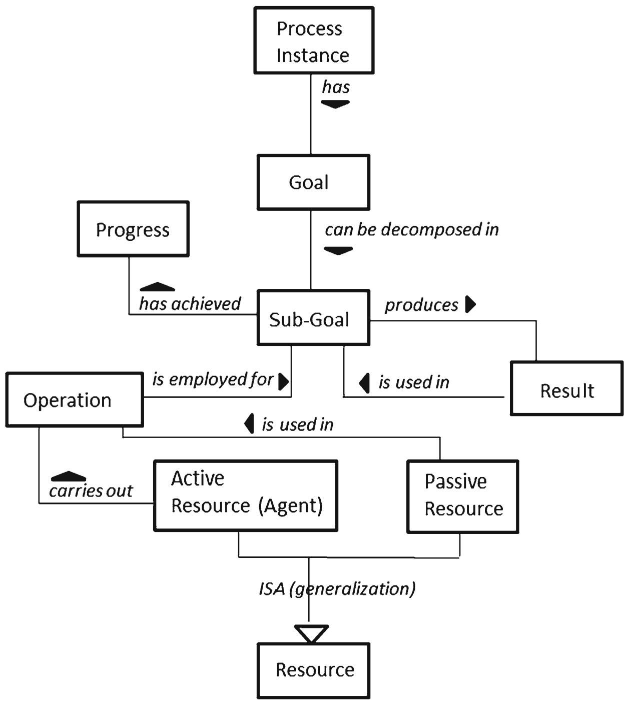

Fig. 2 Concepts and the relationships between them

– pursuing a sub-goal produces *results* that are used when pursuing other sub-goals;
– often, a sub-goal cannot be reached at once, and thus, there is a need for recording the *progress* achieved when pursuing this sub-goal;
– reaching a sub-goal requires resources that can be divided into two categories: *passive*, such as energy or money, and *active* or *agents* that perform actions, such as people or robots;
– to reach a sub-goal, an agent or several agents need to perform one or more *operations* (*oractions/activities*).

The concept introduced above and the relationships between them are illustrated in Fig. 2.

Because we consider the instances of a process that belong to the same type to be similar to each other, we assume that their goals, results, and sub-goals are also similar. This allows us to define meta-concepts for all of the concepts listed above, e.g., meta-goal, meta-result, and meta-sub-goals. A meta-concept requires having a pattern with placeholders (variables) that can be used for any instance. Meta-concepts become parts of the definition of the process type. For example, a meta-goal for the sales process can be expressed in the following:

To sell customer X product Y for the minimum price Z, with budget B for the effort.

[4] See a similar discussion about work systems in [3].

When it is clear from the context that we are discussing business process types, rather than instances, the prefix "meta-" is omitted.

2.3 Process steps

In this research, we assume that a business process under investigation is "somewhat structured." The minimal requirement for the process structure is that a (meta) goal of the process type is decomposable into several (meta) sub-goals and that the pursuit of each sub-goal can be entrusted to different participants or groups of participants in the process. Based on this assumption, we introduce the concept of *process step* as the combination of a sub-goal and the results associated with it, the progress achieved, the participants, and the operations (see Sect. 3.2). The concept *process step* is applied to both the instance level and the type level. On the instance level, a process step represents a particular sub-goal, the result achieved so far, the people engaged in achieving the sub-goal, and the operations—the ones already completed and those that are planned. On the type level, the step represents a sub-goal template, the roles of participants to be engaged, and a template for formatting the result. Graphically, process steps are represented as boxes (rectangles), in the same way that it is done in the systems development process in Fig. 1. This process will be used in the rest of the paper for illustrating the ideas being developed.[5]

3 Research method: design science research

The development of the framework was carried out according to the DSR paradigm. In this section, we introduce and present our view on DSR, which is used to explain the way our solution has been developed.

3.1 Generating and testing hypotheses for adoption

Design science research (DSR) [9,27,36] is related to finding new solutions for problems known or unknown [4]. To be counted as a DSR solution, the solution should be of a generic nature, i.e., applicable not only to one unique situation, but to a class of similar situations, cf. Principle 1 of [40]. There is a substantial difference between DSR methodologies on one hand and widespread qualitative and quantitative methods [34] on the other. The latter are aimed at investigating real-life situations as is, or as they were at some point in the past, in order to find commonalities between them that can give rise to a theory explaining the current or past state of affairs. Focusing on the present and past also allows the use of statistical methods, because information can be gathered on many similar, real-life situations, ensuring that the size of a sample is sufficiently large for their use.

Focusing on the present and past in such a dynamic area as information system (IS) has a major drawback. It means that research follows the industry/practice and explains its successes and failures rather than showing new ways to proceed, as argued in [35]. DSR, with its focus on generic problem solving, tries to overcome this drawback. This kind of research can be considered as an activity aimed at generating and testing hypotheses[6] for future adoption in practice [12,13]. Therefore, implementation and verification of a generic solution [12,13], called an artifact [27,36], in at least one situation, are a critical part of design science. It is usually referred to as a demonstration or proof-of-concept [36]. This stage shows whether a hypothesis is a candidate for adoption or whether it needs to be discarded or improved.

Design science research, on its own, cannot provide sufficient evidence in favor of a hypothesis. It can only demonstrate that a generic solution works in one or several specific situations. A definite proof comes only when and if the industry/practice adopts the solution, generating a sufficient number of examples of its usage in real life, so that standard qualitative and quantitative methods can be employed to prove or disprove the hypothesis generated by DSR. Therefore, DSR cannot be placed in the same category as empirical research, but should be regarded as complimentary. In short, we agree with [27] that DSR represents a distinct research paradigm with a focus on generating hypotheses on how the future could look and carries out an initial filtering of them in order to remove hypotheses that are not worth pursuing. By widening the employment of DSR, the IS discipline can acquire a leading position in the field of practice.

3.2 Movement between two worlds

According to [12,13], DSR, as a way of generating and testing hypotheses for generic solutions, requires researchers to act in two different worlds: (a) the real world of specific problems and solutions in local practices, and (b) the abstract world of generic problems and solutions. The movements of researchers in these two worlds can be visualized, as in Fig. 3.

The upper part of Fig. 3 represents the real world as a specific situation-problem-solution space with three axes: (1)

[5] The simplified software development process from Fig. 1 is used only as an illustration for the concepts being developed in the paper. We are not suggesting and/or arguing for/against any particular method of software system development.

[6] In this paper, we use the term hypothesis in its general meaning: "a supposition or proposed explanation made on the basis of limited evidence as a starting point for further investigation" (Oxford dictionaries: http://oxforddictionaries.com). Our usage of this term bares no connotation as to how it is used in positivists' research methods.

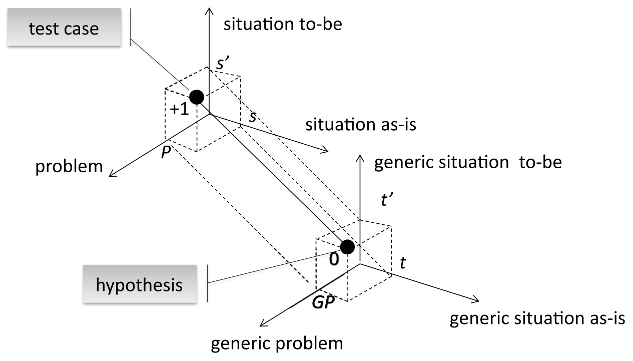

Fig. 3 Design science requires researchers to move between two worlds, the real world and the abstract world

situation as-is, (2) problem, and (3) situation to-be (i.e., a possible solution)[7]. A problem is defined as a set of situations considered to be undesirable from some point of view. For example, the problem—"a lack of communication between the members of our project team"—can be represented as a set of all possible ways of organizing our team so that its members do not exchange important information between them in the frame of the project. The situation, as-is, in this case, is "our team as of today." A possible solution—situation to-be—could be "our team using collaborative software for communication."

A point $<s,\ P,\ s'>$ in the upper space in Fig. 3 is called a test case. A test case represents a question of whether problem P in situation s (i.e., $s \in P$) can be solved by transforming it into situation s' (i.e., $s' \notin P$). The answer to this question can be either negative—transforming s into s' has not solved the problem P (i.e., $s' \in P$), or positive — transforming s into s' has solved the problem P (i.e., $s' \notin P$). In the latter case, s' is considered to be a solution for problem P in situation s. The results of tests can be visualized by assigning the weight +1 to the point $< s,\ P,\ s' >$ if s' is a solution for P in s (see Fig. 3), and otherwise, assigning -1.

The lower part of Fig. 3 represents the abstract world as a generic situation-problem-solution space with three axes: (1) generic situation as-is, (2) generic problem, and (3) generic situation to-be, (i.e., a possible generic solution). We consider that a generic situation (as-is or to-be) is represented by some kind of a template that defines a set of similar situations: the template serving as an extension of this set. The generic problem is defined as a set of situations that are undesirable from some point of view. The generic problem normally encompasses a larger number of situations than a specific situation from the real world. For example, if a specific problem P can be defined as *lack of communication between the members of a specific project team*, the corresponding generalized problem, *GP*, will be defined as *lack of communication in project teams of certain kind*. The correspondence between P and *GP* can be defined through set inclusion:

$P \subset GP$.

A point $< t,\ GP,\ t' >$ in the generic space is called a hypothesis and represents a question of whether a problem $P (P \subset GP)$ in a real world situation as-is s described by template t can be solved by transforming s into situation to-be s' according to template t'. Template t' constitutes an artifact, as it is called in the literature of design science.

The relationships between the two spaces in Fig. 1 are of the type *instantiation/generalization*. If real situation s satisfies template t, then s is referred to as an instantiation of t. Alternatively, t is considered to be a generalization of s. Extending the notion of instantiation/generalization to the points in the two spaces, test case $< s,\ P,\ s' >$ in the specific space is considered to be an instantiation of hypothesis $< t,\ GP,\ t' >$ in the generic space if s is instantiation of t, s' is an instantiation of t', and $P \subset GP$. Alternatively, hypothesis $< t,\ GP,\ t' >$ is called a generalization of test case $< s,\ P,\ s' >$.

The knowledge about hypotheses in the generic space can be visualized by assigning weights to the points of this space according to the following rule:

- +1, when there is a statistically valid proof (many test cases) that t' is a generic solution for generic problem *GP* in generic situation t;
- -1, when there is at least one test case instantiating the hypothesis with weights -1 assigned to it;
- 0, when there is at least one test case instantiating the hypothesis with weights +1 assigned to it, but the statistically valid proof has not been obtained so far.

In terms of the two spaces and weights introduced above, the primary goal of DSR, as a process of generating and testing new solutions for adoption, can be defined as finding "blank" points (without weights) in the generic space and assigning them weights 0 or -1. Assigning weight +1 cannot be considered as a task of DSR, as research itself cannot generate a statistically sufficient number of test cases (as discussed in Sect. 3.1). This assignment can only be made in practice by adopting the solution and producing enough cases for empirical research to investigate. The efforts of DSR are primarily directed at finding the generic solutions that work (weight 0). However, discovering the potential solutions that do not work during this process also constitutes valuable knowledge that should be published and marked on the map of Fig. 3 with weight of -1.

[7] This space includes all past, present, and possible future situation, problems, and solutions. Using orthogonal spaces in Fig. 3 is a simplification that is made in order to use relatively simple pictures to illustrate the basic concepts. In reality, the spaces are neither orthogonal, nor do they have a real metric to measure the distance between the points. For details, see [12,13].

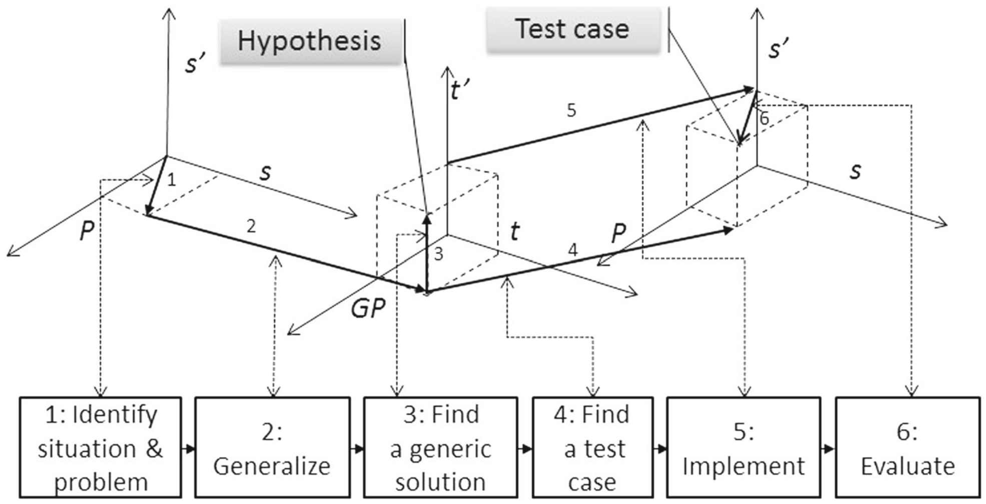

Fig. 4 Generic problem solving

3.3 Using the two-worlds model for conducting a DSR project

Design science research does not impose a specific order of movement inside and between the two spaces introduced in Sect. 3. For example, a researcher can start with a specific space and find an innovative solution for a problem in a specific situation, and then generalize the situation, problem, and solution when moving to the generic space. In this case, the initial solution will automatically serve as a test case of a generic hypothesis. Another possibility is to start with a specific situation and problem, then generalize them, and try to invent a generic solution that can be tested to solve the initial, specific problem in the initial situation. From the opposite side, the researcher can start with designing a generic solution for an "unknown" problem, implement it in some specific situation, and then analyze whether it solves some specific problem, and whether this specific problem and situation as-is could be generalized.

The two-worlds model, depicted in Fig. 3, can be used as a map to help the research team to identify its position in their DSR project and see what alternatives are open for them to continue the project.[8] For the project in focus in this paper, the trajectory of movement in the two worlds can be depicted in the form of Fig. 4. We call this trajectory "generic problem solving."

In generic problem solving, the project starts with *identifying* a specific situation and a problem in it that needs a solution—arrow 1. The next step is *generalizing* the situation and the problem—arrow 2. After that, the *design of a solution* follows— rrow 3. As the specific situation that initiated the project might not be available for testing the new solution, we cannot always rely on it. In such a situation, the next logical step is to find another case, i.e., a specific situation and a problem in it that corresponds to the given generic situation and generic problem. This is represented by arrow 4. The next step is *implementing* the generic solution in this new specific situation—arrow 5. The last task is *evaluating* whether the problem had been solved—arrow 6.

Figure 4 will be used throughout the text of the paper when discussing different stages of our project.

4 From identifying the problem to outlining the solution

4.1 Encountering a specific problem in practice

The specific problem that has initiated this research has been encountered in the practice of a small Swedish consulting company, IbisSoft, to which the first author has been affiliated. The company has developed a tool for building BPS systems delivered as Web-based service. The tool, called *iPB* (iPB Reference manual [28]), is based on the state-oriented view of business processes [30]. *iPB* employs the shared spaces technique for organizing communication/collaboration between process participants [11,14].[9] The shared space of the process is visualized as a simple diagram (called a process map) where each process step is represented by a box. A step box can be expanded into a structured form that reveals the details of this step, e.g., progress or participants.

The *iPB* tool was developed during a pilot project for building a system to support one of the processes in the social

[8] The idea corresponds to Ronald Giere's consideration of theories being like maps [24].

[9] A shared space is an information space that can be accessed by multiple users, such as a blog, wiki or a personal journal. A process shared space is an information space that supports communication/collaboration of process participants in the frame of a process instance.

security office of the Swedish municipality of Jönköping. *iPB* proved to be useful, not only for supporting the original process of the pilot project, but also for other processes in the municipality. It proved to be suitable for supporting the so-called loosely structured business processes, i.e., processes that are controlled by events and information gathered in the course of the processes, rather than by a predefined sequence of operations.

Based on the results from the pilot project, IbisSoft decided to market *iPB* as a generic BPM tool outside the municipality of Jönköping. In this business activity, an issue arose of how to convince a customer that the *iPB* tool was right for the process in question. An additional issue related to the main one was how to differentiate *iPB* from competing products, especially those that were built using the workflow technique.[10] In the terms of Fig. 4, the situation and the problem (arrow 1) when dealing with a specific customer can be defined as follows:

- (*Situation*) given a specific BPM tool, i.e., *iPB*, and a specific customer process, e.g., processing reclamations in a specific organization;
- (*Problem*) how to determine, in a convincing manner, whether *iPB* is a suitable tool for this process.

The first solution to be tested is to solve the problem while remaining on a specific level by developing a functioning prototype for the customer process using *iPB*. This solution, however, has a major drawback: If *iPB* shows to be unsuitable for the process in question, the time invested has been lost.

To eliminate this drawback, IbisSoft decided to change the point of view from the vendor-centered to a customer-centered one, which can be formulated as:

- (*Situation*) given a specific customer process, e.g., processing reclamations in a specific organization, and a specific set of BPM tools, like *iPB* and Apian BPM suite;
- (*Problem*) how to determine which tool in the set is most suitable for this process.

In business terms, IbisSoft decided to develop an independent consulting service to match the needs of the process with the capabilities provided by various BPM tools available on the market. This obviously required generalizing the situation and problem, and finding a generic solution—arrows 2 and 3 in Fig. 4.[11]

4.2 Understanding the problem

As was formulated in the previous section, from the customer point of view, the current problem is as follows.

How to choose a right BPM tool for the given process from the multitude existing in the market?

The direct approach here would be to check each available tool against the given process. There are two issues with the direct approach. Firstly, there are *too many* tools from which to choose. Secondly, it is not a *simple matter* to establish whether a BPM tool is suitable for building a BPS system for a particular business process. There are several sources of the difficulties, as explained below.

To clarify the two issues stated above, we analyze the market of BPM tools delivered as cloud service. This particular market has been chosen because, for the moment, it offers the wider diversity of BPM tools, than the market of traditional BPM tools aimed for local installation. This diversity exists for two reasons. Firstly, all major providers of BPM solutions are moving their BPM tools to the cloud and offering them as services, see, for example, [7]. Secondly, new providers start directly with providing services and bypassing the deployment of tools for local installation, see, for example, [1,37,39].

Based on the analysis of sales material and technical documentation of existing BPM tools[12] we can roughly classify all tools on the basis of two parameters: (1) application domain and (2) BPM paradigm. As far as the application domain is concerned, we can differentiate between domain-independent tools, such as Appian Cloud BPM, ActionFlow, SpringCM or iPB, and domain-dependent services, such as SalesForce or ProjectPlace. As far as the BPM paradigm is concerned, there are several well-established paradigms, for example, workflow or case management, and many others are in making, for example, artifact-based BPM or social BPM.

The differences between domain-independent and domain-dependent tools are not so much in the capabilities they provide, but in the terminology, they use for marketing and sales, and in manuals that explain how to use the tools. For example, [39], which markets itself mainly in the domain of customer relations management (CRM), uses terminology related to sales, e.g., customer profile, account history, decision makers, or marketing campaign. [37] uses project-oriented terminology, such as project planning or project template.

[10] This paper is not aimed at evaluating or promoting *iPB*. A short description of the *iPB* project is included with the aim of demonstrating the DSR approach used in this research, more exactly to show how the research was initiated by a problem in IbisSoft. For the scope of this paper, it is enough to underline that *iPB* does not belong to the mainstream paradigm of workflow thinking, which created difficulties for its promotion. This, in turn, gave an impulse for finding a generic solution for this kind of problem. Note also that the process model built in *iPB* has no resemblance to the one suggested in this paper.

[11] For the time being, IbisSoft has postponed promotion of *iPB*, using it only as an internal tool. This required us to seek other test cases for the evaluation of our suggestions (see the right-hand side of Fig. 4).

[12] We plan to publish the result of this analysis separately as a part of our continuing work on the solution outlined in this paper.

The domain-independent tools use terminology connected to the paradigm to which they attach themselves. For example, Appian Cloud [19], and [1], which belong to the workflow paradigm, define a process as a detailed workflow chart in some visual notation, e.g., BPMN. SpringCM [41], which belongs to the case management paradigm, uses this paradigm terminology when describing a process, e.g., folder, document, or document routing.

Despite the differences in terminology identified above, various domain- and paradigm-dependent BPM tools also provide common capabilities that are independent of the domain and/or paradigm, such as a dashboard or task management. The latter can be considered as an indication that despite the differences in terminology, different BPM tools could provide comparable capabilities that are equally well-suited to a particular business process.

Based on the analysis above, we can conclude that:

– existing tools are described in functional terms of how to use them in practice, and no references are made to the characteristics of business processes for which these tools would be suitable;
– these descriptions use domain-specific or paradigm-specific terminology, which makes it difficult to compare BPM tools that belong to different domains or paradigms.

As a result, it is difficult to choose a BPM tool suitable for building a BPS system for a particular process without trying several different tools in order to determine which one fits best. This is a time-consuming strategy that is not always feasible to implement. For example, process participants might become tired of trying new BPM tools with different terminologies and different limitations. Avoiding such trials, however, has the following risks:

– a BPM tool that seems to be suitable at the first glance is chosen, and the business is stuck with it for a long time, which may negatively affect the business. For example, one has chosen a BPM tool that requires the establishment of a strict order of operations, while the business relies on the initiative of process participants in order to run properly;
– a BPM tool that could suit a process perfectly is removed from the potential candidates because it uses unfamiliar, domain-dependent, or paradigm-dependent terminology.

The question arises:

Is it possible to minimize the risks listed above while avoiding time-consuming trials or, at least, diminishing the number of them by pre-filtration of presumably suitable tools based on the analysis of the process that needs support?

In the terms of Fig. 4, we can reformulate this question as a position in the generic state space defined as follows:

– (*Generic situation*) given a business process x, and a set of BPM tools $Y = \{y_1, \ldots, y_n\}$;
– (*Genetic problem*) how can it be determined, in a time- and cost-effective manner, which tool $y_i \in Y$ is the most suitable for process x?

This position represents the end of arrow 2—generalization—in Fig. 4.

4.3 Analyzing existing sources

Before starting to design our own solution, we have studied both scientific- and practice-oriented literature to see whether there exist solutions that can be adopted and adapted in practice. The existing sources can be roughly classified according to the following scheme.

1. Sources that suggest a method of evaluating of BPM tools based on the capabilities provided by them
   A typical example is [29], which differentiates about twenty capabilities, such as roles-based routing, process rollback, sub-processes, escalation, and exception. Some of these capabilities are relatively paradigm-independent, e.g., roles-based routing, while others are related to the workflow paradigm, e.g., escalation and exception. As the capabilities identified are ranked in general, and not in relation to properties of specific processes, it is difficult to use such methods when deciding on the suitability of a particular tool for a particular process.
2. Sources that concern selection of a BPM tool from a particular paradigm, e.g., workflow
   A typical example of such a source is [47]. The paper presents a detailed list of tools capabilities related to workflow, such as managing forks and joins. This type of works is potentially useful for the task at hand, but not on its first stage, because the authors do not cover the selection over the range of paradigm- and domain-dependent tools. We believe that this type of solutions is useful only after the suitability of the workflow paradigm for the process has been established. Then, one can go into more detail in order to filter potential candidates further, based on a detailed description of the process in a workflow notation.
3. Sources that concern choosing a method/notation/technique for business process modeling
   Typical examples of such work are [26] and [15]. These studies are directed at finding the best method/notation/technique to depict the details of the process in question. They relate the choice to the goal of the model (e.g., BPR, training), properties of the process (e.g., specialization of agents), and the modeling context. Though this kind of work has some ideas that can be useful for the tasks at hand, they are not specifically directed at selecting BPM tools for building a BPS system.

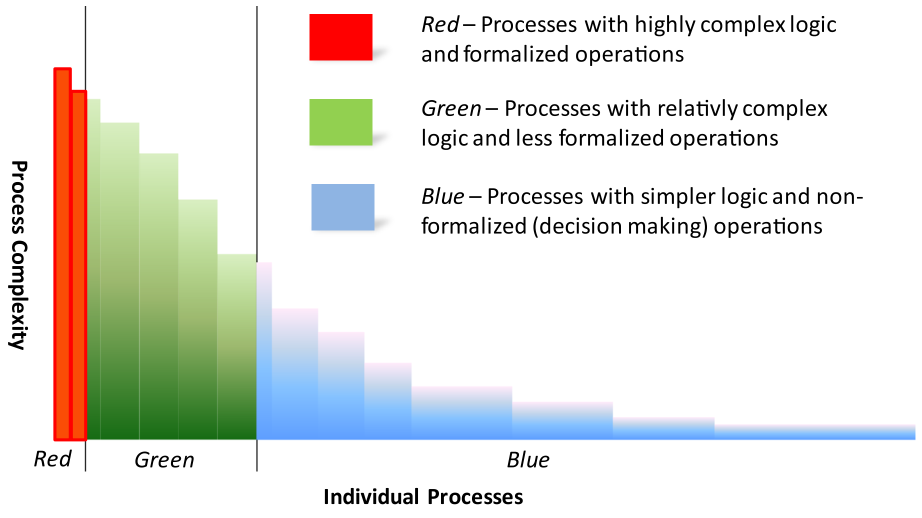

Fig. 5 Distribution of different kinds of processes according to complexity and the number of instances per year in a typical organization, adapted from [23] (color figure online)

4. Sources that describe advantages or limitation of particular paradigms
   The representative example here is [8], which describes those properties of the business process that make it a good candidate to be supported by a BPS system that is built on the workflow paradigm. This kind of works is useful for our goals, but they need to be integrated into a more general solution.
5. Sources that describe unsuccessful usage of BPM paradigms
   Although these works are rare, they do exist. A typical example is [31], which describes an unsuccessful attempt to develop and introduce a workflow-based system in an engineering department. Although this kind of work is useful for our goals, the material from them needs to be converted from the negative perspective of what does not work in a specific situation to the positive one of what could suit the situation better.
6. Sources that consider general characteristics of BPM tools without taking into account the nature of a particular process
   A representative example of this class of papers is [21]. It suggests criteria and requirements on BPM tools of the general type, such as usability (including intelligibility, learnability), scalability, security (including reliability), portability, and presence of modeling notations. Some of these criteria and requirements coincide with the technical capabilities that we investigate in this paper, but most of them do not. The criteria and requirements listed in [21] are definitely of importance and should be taken into consideration in the final selection of a BPM tool. However, they do not give an answer to the question of which BPM tool is suitable for a particular process when the general requirements are satisfied.
7. Sources that take a business perspective on BPM
   A representative example of this class of paper is [6]. It suggests a holistic view on business processes and their management, and highlights business-oriented aspects, such as business network and ecosystem, leadership, people issues, and knowledge, instead of more technical aspects involved in the introduction of software support for business processes. [6] criticize the application of BPM in form of designing a rigid functional structure and taking a mechanistic view on business processes. The value of this kind of papers for our research is that it highlights the issues beyond the flow of operations, such as people and tacit knowledge, which need to be taken into consideration. However, they do not suggest a solution for the question of what type of IT would be suitable for a particular business process.

Summarizing the results of the analysis above, we conclude that to the best of our knowledge, no general solution for selecting a BMP tool for a particular business process can be found in the contemporary scientific- and practically oriented literature. The nearest useful idea for the solution that we found was in the keynote talk of Phil Gilbert [23] at the BPM 2010 Conference. In his presentation, he introduced a simple classification of processes that consists of three categories: red, green, and blue (see Fig. 5), which we found useful for our purposes. Red processes are to be supported by fully-automated transactional systems, green processes are to be supported by workflow-based systems, and blue processes are to be supported by case management and other types of systems aimed at providing flexible support.

Inspired by [23], we started to look for a method that could, with minimum analysis, allow us to determine to which category, red, blue, or green, a given business process belongs. This proved to be a challenging task, which eventually led us to develop the solution presented in this paper.

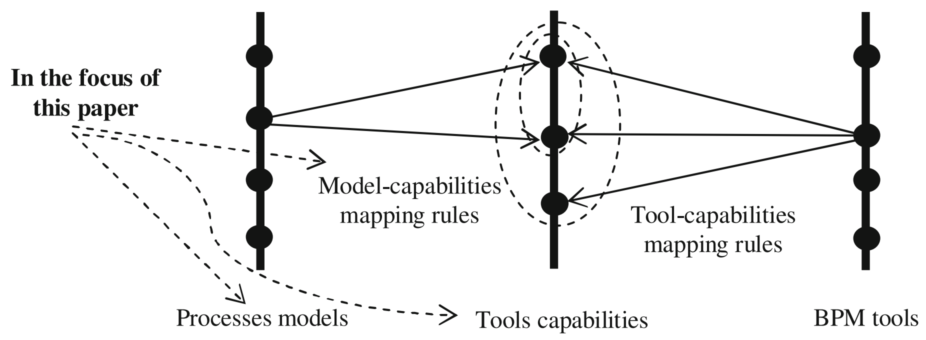
In the focus of this paper
Model-capabilities mapping rules
Tool-capabilities mapping rules
Processes models
Tools capabilities
BPM tools

Fig. 6 The outlined solution explained

4.4 Requirements on the solution

Requirements, in terms of [46], or objectives, in terms of [36], are propositions that narrow down the field of possible solutions. Requirements can be comprised as a list of properties with which the solution should comply, and/or as a list of properties that should be avoided. The latter can be based on known facts that solutions with such properties did not work in the past, or that they did work but had some unintended consequences [12]. In terms of Figs. 3 and 4, requirements are a set of restrictions in the area of search in the "generic situation-problem solutions space" [12].

Below, we list the requirements on our solution, based on an analysis of the problem in the previous sections and our experience related to this problem, as well as our general experience in the field of BPM and common sense.

1. An intermediate abstract level for the presentation of both business processes and tools needs to be introduced to avoid comparing two complex objects: a business process and a BPM tool. On this intermediate level, a process is represented by its "image," which reflects only those properties of the process that are essential for choosing a BPM tool. Similarly, a tool is represented by its "image," which reflects only its capabilities, and disregards the details of its technical implementation. Once created, the image of the tool can be used for matching it with images of any number of business processes. The same is true for the process image: Once created, it can be used for matching with images of any number of BPM tools.
2. Images for both processes and tools should be paradigm- and domain-independent, as we aim to choose a suitable BPM tool that is independent of any specific paradigm or domain.

4.5 Outlining the solution

A generic solution, presented in this section, has been developed based on the requirements in Sect. 4.4 and on our experience of building and marketing BPS systems and BPM tools [5,10,11,14]. The solution consists of five components.

1. A *business process modeling technique* for building high-level business process *models* that are suitable for determining which capabilities are needed for a particular business processes
   The technique should include the following:
   a) a *language* to depict a model;
   b) a practical *methodology* for building a model, e.g., by interviewing process participants and conducting facilitating workshops;
2. A list and descriptions of *capabilities* that can be provided by BPM tools written in a paradigm- and domain-independent form, for example, support for information exchange between the process participants, and enforcement of order of operations performed in the process;
3. *Model-capabilities mapping rules* rules for determining *capabilities* needed based on the given high-level business process model;
4. *Tool-capabilities mapping rules* rules for discovering capabilities provided by a given BPM tool;
5. *Matching algorithm* a procedure that compares a set of capabilities determined for the given process by component 3, with the ones discovered for the available BPM tools by component 4.

The main idea behind the solution is illustrated by Fig. 6. The solution works according the following steps:

1. Each tool is pre-mapped into a set of *capabilities* (component 2) using *tool-capabilities mapping rules* (component 3). The set of capabilities compiled for a particular tool constitutes the image of this tool (see Sect. 4.4);
2. For a business process in question, a high-level business process model is built using *business process modeling technique* (component 1). This model is then mapped into a set of capabilities using *model-capabilities mapping rules* (component 3). This set represents the image of the process (see Sect. 4.4);
3. Comparing the sets of capabilities for the given process and the available tools, the *matching algorithm* (component 5) determines the most appropriate tool for the given business process.

In this paper, we present only the following components of the solution: 1a, 2, and 3. We start with a definition of the language for building a high-level business process model in Sect. 5, which is the main contribution of this paper. Then, we present a list of capabilities (Sect. 6) and guidelines for mapping a high-level process model into capabilities (Sect. 7). The components 1b (methodology of building models), 4 (tool-capabilities mapping rules), and 5 (matching algorithm) are still under development. However, some elements of component 1b are presented in "Appendix 11".

5 A step-relationship business process model

5.1 Why do we need a new modeling technique?

As follows from the solution outlined in Sect. 4.5, the basic idea behind a suggested approach to selecting a BPM tool is (a) building a model of business process in question, and (b) deriving requirements on a suitable BPM tool through analyzing this model. For a model to fit our purpose, the following two conditions, already mentioned in the introduction, should be satisfied:

1. The process model should cover different perspectives and their interconnection, not just presenting one view on the business process, e.g., operational view (workflow), or coordination view. Focusing on a specific view enlarges the risk of missing the requirements that come from another view, or from several views when considered together.
2. The level of details in the model should be just sufficient for gaining an understanding of the kind of tools that would be appropriate for this process, but no more. Going into too many details will require much time for building a model, thus making tool selection a tedious and costly task, especially considering the risk that a selected tool may require building a completely new process model with the notation accepted within this tool later.

To the best of our knowledge, there is no business process modeling technique that satisfies both requirements above. As a rule, a technique covers one major view, though it might have additional features that add some aspects from other views. For example, BPMN [19] is concentrated on the operational view—time-related sequence of operations inside the process, while paying minor attention to data and information flow. Other examples are as follows: IDEF0 [22] presents a functional view; the state-oriented approach from [30] is focused on goals and data; etc. In addition, the existing techniques provide features for depicting details of a process, which may lead the investigator to the path off adding more and more details to the model, while remaining in the same view on the process.

The modeling technique presented in the following subsections is aimed to overcome the drawbacks of the existing techniques in satisfying the two requirements above. Note that it is not aimed to substitute the existing techniques, and it has its own limitations, e.g., of not being suitable for depicting details that, for example, are needed for building a BPS system.

5.2 Basic elements of the step-relationship process model

We consider a high-level business process model as consisting of the elements of two types (syntactical units of the modeling language).

– *Step* a composition of sub-goal and other concepts related to it, as defined in Sect. 2.3
  In business terms, a step represents a work package to be completed in the process. The latter definition, as more understandable for business people, is used when presenting the model for nonresearchers. Each step in a model has a unique name.
– *Relationship* a connection between two steps
  Relationships are typed, and there can be more than one relationship between a pair of process steps, with different relationships belonging to different types. Dependent on the type, a relationship can be symmetrical or asymmetrical. In case of asymmetrical relationships, there can be up to two relationships of an asymmetrical type between a given pair of steps.

We introduce two ways of depicting a model for a particular business process: graphical and matrix.

– In the graphical form, the model consists of a set of diagrams, one for each type of relationships. In a diagram, steps are represented as rectangles (boxes) that have their steps names, as labels, inside them. Relationships are represented as lines between the step boxes (see Fig. 7 for an example).
– In the matrix form, the model consists of a set of square matrices, one for each type of relationship, where both columns and rows correspond to the process steps. Intersection between a row and a column in a matrix shows a relationship between the two steps. The type of content in the cells depends on the relationships type.

In this paper, we mainly use the matrix form for depicting a model, as it is easier to operate with matrices in a formal way. Some relationships are basic, i.e., they are determined when building a model. Other relationships are derived to be used by *model-capabilities mapping rules*. A derived-relationships matrix is obtained by transforming a matrix

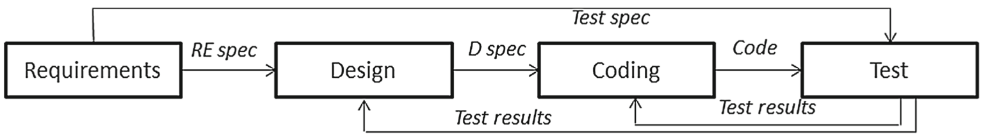

Fig. 7 Graphical representation of Table 1

of one of the basic relationships, or by merging two or more other matrices.

We assume that the number of steps chosen for building a high-level model is rather small (under ten, and more likely to be five or six), so that the whole model is compact in both its matrix and graphical forms. Having a small number of steps means that each of them represents a rather large work package.

A detailed description of the relationships matrices is presented in the subsections below. While describing the matrices, we use the example of the software development process from Section 2 (Fig. 1). The example is intentionally simplified to focus on the semantics of the relationships rather than on the particularities of a real-life process.[13] We have chosen this example for two reasons. Firstly, it allows us to avoid using a formal example with steps named A, B, C, D, etc., because this is usually difficult for a reader to follow. Secondly, this example refers to the domain of the journal, which can help the readers of this article to understand the model we suggest.

5.3 Input–output

The *input–output* matrix shows dependencies of one step on the results achieved in another. A cell (a, b) in the matrix, where a refers to a column and b to a row, specifies what result (i.e., output) from step a (if any) is used as input to step b. In addition to the name of result, a cell can be marked with an asterisk (*), which means that the result is required for step b to be started the first time. An example of an input–output matrix for the process in Fig. 1 is presented in Table 1. In Fig. 7, the input–output dependencies are presented in the graphical form.

The presence of a symmetric pair of nonempty cells (*coding, test*) and (*test, coding*) in Table 1 points to a loop in the execution of the steps, i.e., return from *test* to *coding* (see Fig. 7). To make all loops explicit, we can take the "transitive closure" of the *input–output* matrix creating a derived matrix called the *transitive input–output* matrix (see Table 2). In this matrix, cell (a, b) is marked with a cross 'x' if (a, b) is nonempty in the *input–output* matrix, or there is a sequence of steps $c_1, \ldots, c_n$ such that cells

13 A more realistic example of software development is presented in Sect. 8.2.

Table 1 Example of input–output relationships

Input–output	Requirements	Design	Coding	Test
Requirements				
Design	*Requirements specifications			Test results
Coding		*Design specifications		Test results
Test	*Test specifications		*Code	

Table 2 The transitive input–output matrix derived from Table 1

	Requirements	Design	Coding	Test
Requirements				
Design	x		x	x
Coding	x	x		x
Test	x	x	x	

$(a, c_1), (c_1, c_2), \ldots, (c_{n-1}, c_n), (c_n, b)$ are nonempty in the *input–output* matrix.

In Table 2, there are two pairs of symmetric nonempty cells (*coding, test*), (*test, coding*), and (*design, test*), (*test, design*). The second pair points to the loop going from *design* to *test* via *coding* and returning to *design,* in a situation in which the requirements are not satisfied.

5.4 Parallel execution

The *parallel execution* matrix shows whether two steps may be executed in parallel. If ongoing activity inside step a does not totally forbid carrying out activity in step b, then both cells (a, b) and (b, a) are marked with "x" (the matrix is symmetrical). If none of the steps can run in parallel, the parallel execution matrix will be empty. This will be the case if the system development process in our example runs according to the waterfall fashion.

Suppose that our systems development process template now allows some degree of parallelism to meet hard project deadlines: the step, *requirements,* is partially allowed to run in parallel with both *design*, and *coding*, meaning that the test specifications from the requirements team continue to be prepared while the design and coding are already in progress. Such a case is depicted in Table 3.

Table 3 Example of a parallel execution matrix for a process with some degree of parallelism

	Requirements	Design	Coding	Test
Requirements		x	x	
Design	x			
Coding	x			
Test				

Table 4 Parallel execution matrix for agile system development

	Requirements	Design	Coding	Test
Requirements		x	x	x
Design	x		x	x
Coding	x	x		x
Test	x	x	x	

In a process designed for very tight deadlines, all steps can be allowed to run in parallel. One starts designing as soon as basic requirements are gathered and starts coding when some "implementable" part of the design has been completed. Such a course of action may require a lot of redoing, but it could be the only possibility if there is no way to extend the length of the project. This approach may succeed, provided that the systems development team is experienced and accustomed to working in an agile fashion [2]. A parallel execution matrix for an agile process will look like Table 4.

5.5 Parallel dependencies

By combining the *input–output* matrix with the *parallel execution* matrix, we can obtain a new view on the complexity of a business process. Table 5 is produced by merging Tables 1 and 3 according to a simple rule: Cell (a, b) gets crossed in the new table only if the cell is nonempty in both the input–output matrix and the parallel execution matrix. We will refer to the merged matrix as a *parallel dependencies* matrix. The cross in a cell (a, b) in this matrix means that steps a and b can run in parallel at the same time, because b is dependent on the result from a. In Table 5, there is only one cell that is crossed, (*requirements, design*), which means that the steps *design* and *requirements* can run in parallel, while *design* depends on the results from *requirements* (see the deliberations in Sect. 5.4). If, however, we merge Table 1 with Table 4 (agile software development), we will get many more crosses in the *parallel dependencies* matrix (see Table 6).

A cross in cell (a, b) of the *parallel dependencies* matrix requires special attention as it warrants tight coordination between these steps. Otherwise, the work done in step b may need to be totally redone after substantial changes in the result from step a. Even tighter cooperation is required when both cells (a, b) and (b, a) are crossed.

Table 5 Parallel dependencies matrix that correspond to Table 3

	Requirements	Design	Coding	Test
Requirements				
Design	x			
Coding				
Test				

Table 6 Parallel dependencies matrix that correspond to Table 4 (agile software development)

	Requirements	Design	Coding	Test
Requirements				
Design	x			x
Coding		x		x
Test	x		x	

Sometimes, a cross in the *parallel dependencies* matrix appears because the steps we have chosen are too big. In this case, we can try to remove the parallel dependencies by decomposing (splitting) the original steps into smaller ones. For example, we can split *requirements* into two steps: *specifying requirements* (SR) and *specifying requirements tests* (SRT) (see Fig. 8). Then, the input–output matrix may take the form of Table 7, and the parallel execution matrix will take the form of Table 8. As a result, the *parallel dependencies matrix* becomes empty.

Note that it is not always possible to obtain an empty *parallel dependencies* matrix through decomposition (see the deliberation in Sect. 5.4). Decomposition may not remove all parallel dependencies, or it can introduce new dependencies instead of the old ones. This, for example, happens if we allow steps *SR* and *SRT* run in parallel.

5.6 Weak dependencies

Cell (a, b) in the *weak dependencies matrix* shows whether step b may require something more than the formalized result from step a, for example, a historical trace of how the result has been achieved. For example, it is not unusual for the designers to need more information than exists in the formal requirements. They might need to understand the rationale behind one or more requirements, or need some other background information. Cell (a, b) in this matrix specifies the kind of information from step a, which might be needed to complete step b. An example of this type of matrix for our systems development process is given in Table 9.

The concept of *weak dependencies* reflects the needs for informal communication in the frame of a process instance. It is not always possible to include everything that might be needed for the next step in the formal results, because different process instances may require completely different

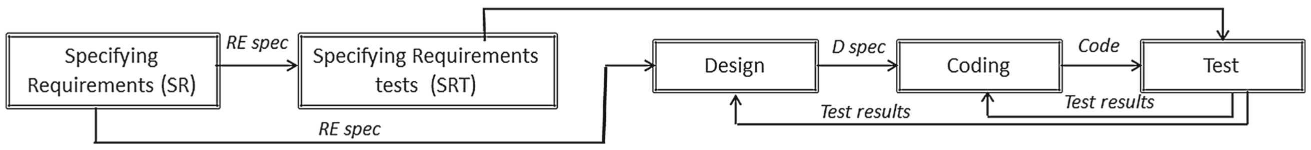

Fig. 8 Graphical representation of input/output relationships after splitting

Table 7 The new input–output relationships matrix

	SR	SRT	Design	Coding	Test
SR					
SRT	*Requirements specifications				
Design	*Requirements specifications				Test results
Coding			*Design specifications		Test results
Test		*Test specifications		*Code	

Table 8 The new parallel execution matrix

	SR	SRT	Design	Coding	Test
SR					
SRT			x	x	
Design		x			
Coding		x			
Test					

Table 9 Example of weak dependencies

	Requirements	Design	Coding	Test
Requirements				
Design	Rational behind requirements communication with the customer			
Coding		Clarification of diagrams		
Test				

information. It is better to start looking for this information on a demand basis, i.e., when there is a need for it. In short, the *weak dependency matrix*, which represents the "on-demand" information flow, complements the input–output matrix to give the full view on the information exchange in the process.

5.7 Teams and their relationships

The *teams* matrix shows the presence of collaborative teams and their relationships. The presence of teams is shown in the diagonal of the *team*s matrix: Cell (a, a) is marked with a light gray color if the team for step a consists of more than one person. The nondiagonal elements show whether the teams participating in different steps intersect. If the teams for steps a and b intersect but do not coincide, we mark both cells (a, b) and (b, a) with light gray. If the teams coincide, we mark these cells with dark gray.

An example of a *teams* matrix for our systems development process is shown in Table 10. Here, we assume that each step does have a team; *requirements* and *design* teams intersect but do not coincide; *coding* and *test* teams coincide.

The diagonal of the *teams* matrix identifies steps that may require support for intra-step collaboration, which is discussed in Sect. 8. The nondiagonal part of the matrix is used for analyzing the need for support of inter-step coordination/collaboration, which is discussed in Sects. 5.8 and 7.

5.8 Inter-step collaboration

By merging the *weak dependencies* matrix (Sect. 5.6) with the *teams* matrix (Section 5.7), we obtain an insight into the need for inter-step collaboration. The result of the merger of matrices in Tables 9 and 10 is presented in Table 11. In this table, one nonempty cell (*requirements, design*) has a light gray background, while the other one (*design, coding*) has a white background. In the first case, *requirements* and *design* teams intersect; thus, additional information from one step to another can be carried out tacitly via intersecting members. In the second case, the *design* and *coding* teams do not intersect; thus, there is a need to make information (other than formal design documentation) that might be needed from *design* for *coding* available on demand. Considering that the design team can be dissolved before coding starts, or be not easily available, this issue needs special attention.

6 Identifying the capabilities of BPM Tools

In this section, we discuss capabilities that can be found in a BPM tool. According to the solution outline from Sect. 4.5, the capabilities of the tools should be defined in a domain- and paradigm-independent way.[14] Their capabilities can be provided separately or in a group, from which one capability cannot be used without the others. A list of the basic capabilities that we believe could and should be expected from BPM

[14] The term capability here is understood as ability to provide support for certain aspect of running business process instances.

Table 10 Example of the teams matrix

	Requirements	Design	Coding	Test
Requirements				
Design				
Coding				
Test				

Table 11 The weak relationships matrix merged with the teams matrix

	Requirements	Design	Coding	Test
Requirements				
Design	Rational behind requirements Communication with the customer			
Coding		Clarification of diagrams		
Test				

tools is presented below. The list has been compiled based on (a) our experience of developing BPM tools, and BPS systems and services, (b) analysis of BPM tools from various vendors, and (c) review of the existing literary sources (see Sect. 4.3). The list is not comprehensive, as it only includes the capabilities the needs for which can be derived from the content of the matrices in Sect. 5. When making the final choice of a BPM tool, other considerations discussed in the literature overview in Sect. 4.3 can be used to support the decision, e.g., usability or security [21]. Our current list of capabilities includes the following.

1. *Information logistics support* (ILS) is aimed at providing process participants with all the information they need to complete their work without being overwhelmed by the details that are not relevant. ILS is particularly important for steps in which the inputs and outputs constitute information objects, such as documents, program code, and test protocols. ILS can be provided in two different ways:
   - by actually sending the results to the next step team, e.g., via email, which we refer to as *conveyor belt logistics* [11];
   - by providing a shared space where the results are stored and made available for the participants of the "next step," which we refer to as *construction site logistics* [11].

   The ILS capability can also provide version control for the information objects that are produced more than once. Version control is easier to achieve by using the construction site logistics than those of the conveyor belt.
2. *Intra-step collaboration support* is aimed at providing a team working on the same step with the means to store/retrieve intermediate results and communicate with each other synchronously and/or asynchronously.
3. *Inter-step collaboration support* is aimed at providing the teams, or individuals working on different steps with the means to access intermediate results obtained in each other's steps and communicate between the teams synchronously and/or asynchronously.

   Note that intra-step and inter-step collaboration may require different means of support. In the first case, the communication can be between people of the same profession who reside in the same department. In the second case, the communication can be between people of different professions who reside in different departments.

   Note also that term collaboration in this paper is used in a special way. It does not cover cases of accepting inputs from the previous steps, or forwarding outputs to the next steps. The latter two cases are considered as belonging to the ILS issues.

4. *Process flow restrictions enforcement* ensures that the rules establish for the process flow are strictly followed. Examples of such rules include the following:
   - ensure that the steps that cannot run in parallel run in turns;
   - ensure that if a step needs a result (output) from some previous step it waits until the latter step is finished.
5. *Process flow support* ensures smoothness of the process flow; it ensures that the steps that can be activated (i.e., inputs are ready) are activated at once. This, for example, can be done by informing process participants that they should start working on their step by sending them the required inputs (when the conveyor belt ILS is employed), or notifying them that the inputs have been placed in the shared space designated for them (when the construction site ILS is employed).
6. *Participation restrictions enforcement* ensures that "right" people are participating in various process steps. The rules can be established because of external legislation (e.g., Sarbanes-Oxley) or can be decided upon internally. The rules concern, for example, who can participate in which steps, which information is available to each kind of participant, and whether the step teams can intersect.
7. *Resource assignment support*. This capability means automatic or semiautomatic formation of step teams

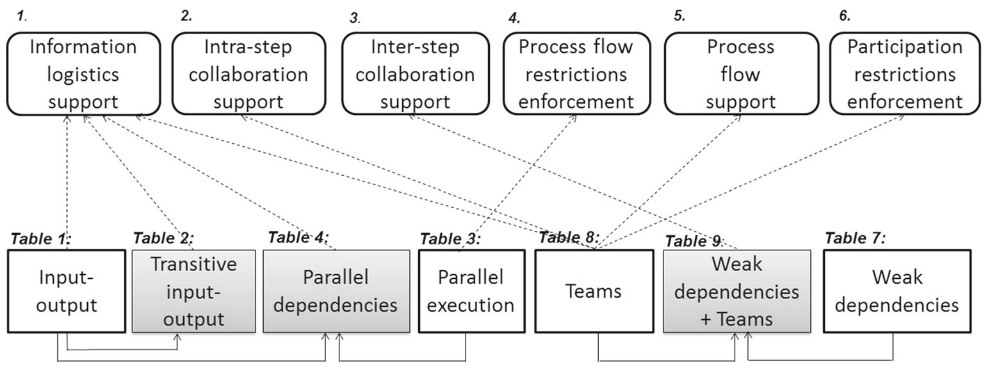
1.
Information logistics support
2.
Intra-step collaboration support
3.
Inter-step collaboration support
4.
Process flow restrictions enforcement
5.
Process flow support
6.
Participation restrictions enforcement
Table 1:
Input-output
Table 2:
Transitive input-output
Table 4:
Parallel dependencies
Table 3:
Parallel execution
Table 8:
Teams
Table 9:
Weak dependencies + Teams
Table 7:
Weak dependencies

Fig. 9 Matrices usage for identifying capabilities

based on qualifications and availability of process participants.

8. *Support for domain-specific operations* includes tools to complete the operations inside the step, such as compiling or testing a program. These tools can be general, such as an office package (MS or OpenOffice) or specialized, such as compilers for specific languages.

7 Mapping a model into capabilities

The easiest way to choose a BPS service is when the process is "workflowable" in a high degree [8]. With the help of our high-level business process model, "workflowability" can be defined as follows:

- the *parallel execution* matrix is empty—dependent steps are executed in turns;
- the *teams* matrix is empty—one and unique person per step;
- the *weak dependencies* matrix is empty (only formalized inputs are relevant for steps execution).

In this case, any BPM tool that provides a workflow solution, e.g., [7], would be suitable.

In a situation in which "workflowability" cannot be established and cannot be obtained by decomposition of the steps (see Sect. 5.5), each capability in the list in Sect. 7 needs to be considered separately against the properties of the business process in question. Figure 9 shows which matrices from Sect. 5 can be used to determine the needs for which capabilities. In addition, the bottom part of Fig. 9 shows which matrices are basic—white background, and which are derived —gray background. Arrows between matrices show which basic matrices are used to produce the given derived matrix. Note that in this paper, only capabilities 1–6 of the list from Sect. 6 are covered.

The rules for choosing BPS service capabilities are as listed below (see Table 12 for a summary of the rules).

1. *Information logistics support.* The capability is desirable as long as the *input–output* matrix (Table 1) identifies results in the form of information objects that need to be passed between the steps. It is less critical when the *teams* matrix (Table 10) identifies that the step teams intersect. In this case, the responsibility of moving the results from one step to another could be assigned to the intersecting members of the step teams and be completed outside the frame of a BPS system. If the teams coincide, there is even less need for information logistics support.
   In cases in which the *transitive input–output* matrix (Table 2) shows that there are no iterative loops (no symmetric nonempty cells in it) and the *parallel dependencies* matrix (Table 5) is empty, the conveyor belt information logistics will work satisfactorily. When loops are present, there can be many versions of the same information objects. When these versions are just sent from one step to another, there is a risk that a wrong version will be used instead of the right one. Having a shared space (construction site ILS), in which a new version substitutes the old one completely, would be preferable in case of loops. Having version control for such shared spaces can provide additional advantages if one needs to understand the difference between the old and new versions.
   A nonempty *parallel dependencies* matrix (Table 5) requires even more attention to information logistics. This is especially so when the teams of two steps with parallel dependencies neither coincide nor intersect (Table 10). In this case, any new piece of information should also include an explanation about whether it is a complement to what has already passed, or a substitution of the old piece, or both. Having dedicated shared space with version control and explanatory comments would constitute appropriate information logistics support in this case.
2. *Intra-step collaboration support.* A capability is desirable when the *teams* matrix (Table 10) identifies steps that have teams (cells marked by the light gray background in the diagonal of the matrix). The capability

Table 12 Model-capabilities mapping

Condition	Requirement
Information logistics support	
IF *input–output dependencies* between steps do exist (nonempty cells in Table 1) AND some step *teams* do NOT *intersect* OR *coincide* (no light or dark gray colors in some nondiagonal cells in Table 10)	*Information logistics support* is required
IF all step *teams* do *intersect* (light gray in nondiagonal cells in Table 10)	*Information logistics support* may NOT be required
IF all step *teams* do *coincide* (dark gray in nondiagonal cells in Table 10)	*Information logistics support* may NOT be required
IF *loops* do NOT exist (no symmetric nonempty cells in Table 2) and *parallel dependencies* do NOT exist (all cells are empty in Table 5)	*Conveyor belt information logistics* is suitable for providing *Information logistics support*
IF *loops* do exists (there are symmetric nonempty cells in Table 2)	*Construction site information logistics (i.e., shared spaces)* is more appropriate for providing *Information logistics support*
IF *parallel dependencies* do exist (nonempty cells in Table 5) AND step *teams* for which the parallel dependencies exist do NOT *intersect* OR *coincide* (no gray colors in respective cells in Table 10)	*Construction site information logistics (i.e., shared spaces)* is more appropriate for providing *Information logistics support*
Intra-step collaboration support	
IF step *teams* do exists (light gray color in some diagonal cells in Table 10)	*Intra-step collaboration support* is required
Inter-step collaboration support	
IF *weak dependencies* do exist (nonempty cells in Table 9)	*Inter-step collaboration support* may be required
IF *weak dependencies* do exist AND the step *teams* with dependencies do NOT intersect (white background in nonempty cells in Table 11)	*Inter-step collaboration support* is required
Process flow restrictions enforcement	
IF steps do NOT run in parallel OR only few steps do run in parallel (i.e., empty or sparse Table 3)	*Process flow restrictions enforcement* is required
Process flow support	
IF step *team* do NOT *intersect* OR *coincide* (no light or dark gray colors in nondiagonal cells in Table 10)	*Process flow support* is required
Participation restrictions enforcement	
IF step *teams* do NOT coincide (no dark gray colors in nondiagonal cells in Table 10)	*Participation restrictions enforcement* may be required

should allow storage and sharing of the intermediate results. In addition, it may include messaging and online communication, such as chat, voice, or teleconferencing. The basic needs can be solved by a shared space structured according to the needs of the team, with or without version control capabilities. Such a space can include forums for discussion and journals to record the internal or external events, for example, communication with customer/supplies. A step shared space can be useful even when a step "team" consists of one member (the white background in the diagonal of the *teams* matrix). This is true when he/she cannot complete the whole step in one go and needs to return to the step several times before it is completed.

3. *Inter-step collaboration support.* The needs for this capability can be identified by the merged *week dependencies* + *teams* matrix (Table 11). The capability is desirable when there are nonempty elements in the matrix. The needs for this kind of support are even greater if the teams for steps with weak dependencies do not intersect—nonempty cells in the matrix with a white background. One way of arranging such collaboration is by having a communication channel between the teams. A dependent step team can send a request for extra information and get it back through the same channel. This will work if the members of the team which have the information are still available for questioning.
Another way of arranging inter-step collaboration is based on the shared spaces technique employed for intra-step collaboration. If the step shared space is made accessible to the team of a dependent step, the members of the latter can, themselves, find the information they need. This will work provided that the shared spaces are structured in a way that makes it easy to navigate in them, even for the process participants who do not participate in the steps from which they need to obtain the information.
4. *Process flow restrictions enforcement*. This capability is desirable if the *parallel execution* matrix (Table 3) is

empty or sparse. If many steps can and should run in parallel, this capability will not be particularly useful.

5. *Process flow support.* This capability is very useful if the *teams* matrix (Table 10) shows that steps teams neither intersect nor coincide. However, it does not do any harm to have it, even if they do intersect. In the situation in which many steps can run in parallel and steps teams do not intersect, a more sophisticated coordination mechanism is required than just process flow support (see the ILS-related discussion above).
6. *Participation restrictions enforcement.* This capability might be needed if steps teams do not coincide, which can be easily determined from the *teams* matrix (Table 10). The actual need for this capability depends on the reasons that are not revealed by the step-relationship model. If conveyor belt information logistics is employed, the restrictions are established by sending results only to the participants of the steps in which these results are to be used. If shared spaces are employed for information logistics support, the participation restrictions are realized by limiting access to some parts of shared spaces.

The rules above are summarized in Table 12 in a quasi-formal way.

8 Demonstration and evaluation

Demonstration, or proof-of-concept [36], is the application of a generic solution (or artifact in terms of [36]), designed by researchers to a real-life situation in order to test its feasibility. In terms of Fig. 4, demonstration encompasses the following: (a) finding a specific case that instantiates the given generic situation and problem—arrow 4; (b) implementing the generic solution in this specific situation— arrow 5; (c) evaluating the results—arrow 6.

The solution suggested in this paper naturally consists of three parts: (a) building a step-relationship model, (b) analyzing the model and deriving requirements on a BPS system, and (c) choosing tools for building the system. A project that can produce a test case for full evaluation of the suggested method therefore would extend from process modeling to implementing a BPS system in practice in the organization in question. We can then evaluate how well the implemented system satisfies the needs of process participants. Such a project for a modestly complex process may take a number of years to complete. While looking for an opportunity to complete such a project, we have tested parts of the solutions on less ambitious projects. One type of such tests concerns the step-relationship model itself and is aimed to answer the following question:

Is it possible to build such a model for a real business process, and will this model be understandable and practically useful, even if the usage does not lay in selecting tools for BPS system development?

This question corresponds also to the main focus of this paper, which is the new modeling technique itself. This section answers the question positively by presenting two cases that include building step-relationship models of real business processes. The cases themselves are presented in Sects. 8.1 and 8.2, while Sect. 8.3 discusses lessons learned from the projects presented.

As we already mentioned in the introduction, DSR on its own cannot provide sufficient evidence in favor of a hypothesis. Definite proof comes only when and if the industry/practice adopts the solution, which is the reason for [36] to consider the dissemination of results as an important step in a DSR project. Therefore, in Section 8.4, we discuss the measures to be undertaken for facilitating the adoption of the step-relationship modeling technique.

Note also that step-relationship modeling is not the only part of the suggested solution that can be evaluated on its own. The other part that can be evaluated independently concerns an analysis of the capabilities provided by BPM tools. The challenges related to the development of this part are discussed in Sect. 9.2.

8.1 Process of preparing a course at a university

Our first test case concerns a process of preparing and giving a course at a university. The course process consists of giving the course as well as its preparation and evaluation by the teacher and the students. Five major steps were identified in the course preparation process.

1. *Plan course* includes a number of meetings with involved teachers to decide which teaching and learning activities to carry out during the course as well as their sequence. The step also includes deciding and producing teaching material for the course. Finally, an evaluation form needs to be designed, which will be filled out by the students when the course has ended.
2. *Schedule course* consists of composing a schedule with dates, times, and locations for the lectures, lessons, and seminar as well as date, time, and location for the written exam and other teaching and learning activities. The step includes a number of interactions between the teacher responsible for the course and the person responsible for scheduling courses in the department.
3. *Publish course material* consists of printed course material. The printing is done by the person responsible for printing.
4. *Learn and teach* includes a number of teaching and learning activities, such as lectures, lessons, and seminars, managing assignments, and carrying out exams. It also

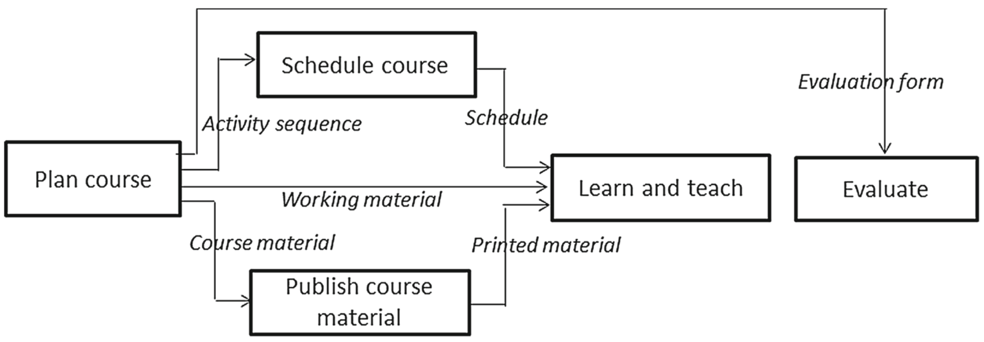

Fig. 10 A course preparation process at a university

includes giving feedback on and/or grading reports and exams.

5. *Evaluate* includes the students evaluating the course after the end of the course. The step also includes an analysis of the evaluation carried out by the teacher responsible for the course.

The step-relationship matrices we created for the course process are presented in "Appendix 10". The graphical representation of the input/output relationships between the steps is illustrated in Fig. 10.

Analysis of the model of the course process with the help of rules from Sect. 7 resulted in the following recommendations on BPM tools capabilities.

1. *Information logistics support*. The *input–output matrix* from "Appendix 10" shows that there are many information objects that need to be passed between the steps. According to the *teams matrix* from "Appendix 10", most step teams intersect, and thus, this property might not be critical. However, due to the pure number of information objects to be passed, it will be difficult to manage all of them manually. The *transitive input–output matrix* from "Appendix 10" indicates that there are no loops for formalized input–outputs. The *parallel dependencies matrix* from "Appendix 10" is far from sparse (five crosses), which means that the conveyor belt approach to information logistics is not particularly suitable. Summarizing the above, the course process would benefit from information logistics support, based on the construction site approach, e.g., shared spaces techniques.
2. *Intra-step collaboration support*. The intra-step collaboration is desirable, according to the *teams matrix* from "Appendix 10". As shared spaces techniques is recommended for this process (see 1. above), shared spaces could also be used for intra-step collaboration.
3. *Inter-step collaboration support*. The *weak dependencies + team matrix* from "Appendix 10" is not empty, which shows that inter-step collaboration takes place. However, the majority of nonempty elements in this matrix have a light gray background. The latter shows that the inter-steps collaboration is not critical and can be solved by intersecting teams.
4. *Process flow restrictions enforcement*. The *parallel execution matrix* from "Appendix 10" shows that most activities can be carried out in parallel. Therefore, process flow restrictions enforcement could hardly apply.
5. *Process flow support*. The *teams* matrix from "Appendix 10" shows that most steps teams intersect. Therefore, process flow support is not a critical capability here, but it could be useful if provided.
6. *Participation restrictions enforcement*. As the *teams matrix* from "Appendix 10" shows that none of the teams coincide, the needs for participation restriction should be investigated based on the other considerations than matrices. As we clearly have different participants groups, i.e., teachers and students, this capability is needed, and it should be expressed in different rights of viewing and updating shared spaces introduced for information logistics support and intra-step collaboration.

The model in Fig. 10 and "Appendix 10", and the analysis above have been used for choosing a modeling technique for building a detailed model of the course process and a prototype of a system to support it. The aim of the detailed model is to solicit detailed requirements for a system, including support for information flow and participants' collaboration. As the analysis above pointed to nonworkflow ability of the course process, we looked for modeling techniques and tools that were not based on workflow thinking, in particular techniques and tools that support data-centric/data-driven [33], or artifact-driven [20] process modeling. After choosing a tool that satisfied requirements and was available, we produced a detailed data-centric process model and a working prototype of a BPS system for the course process [17]. The run-time environment of the tool used for modeling ensured functioning of the prototype based on the interpretation of the model without much extra effort for building the prototype. The only effort that required was fine tuning of the Web forms.

The model and the prototype were built by a MS student who used the authors as domain specialists. Both were

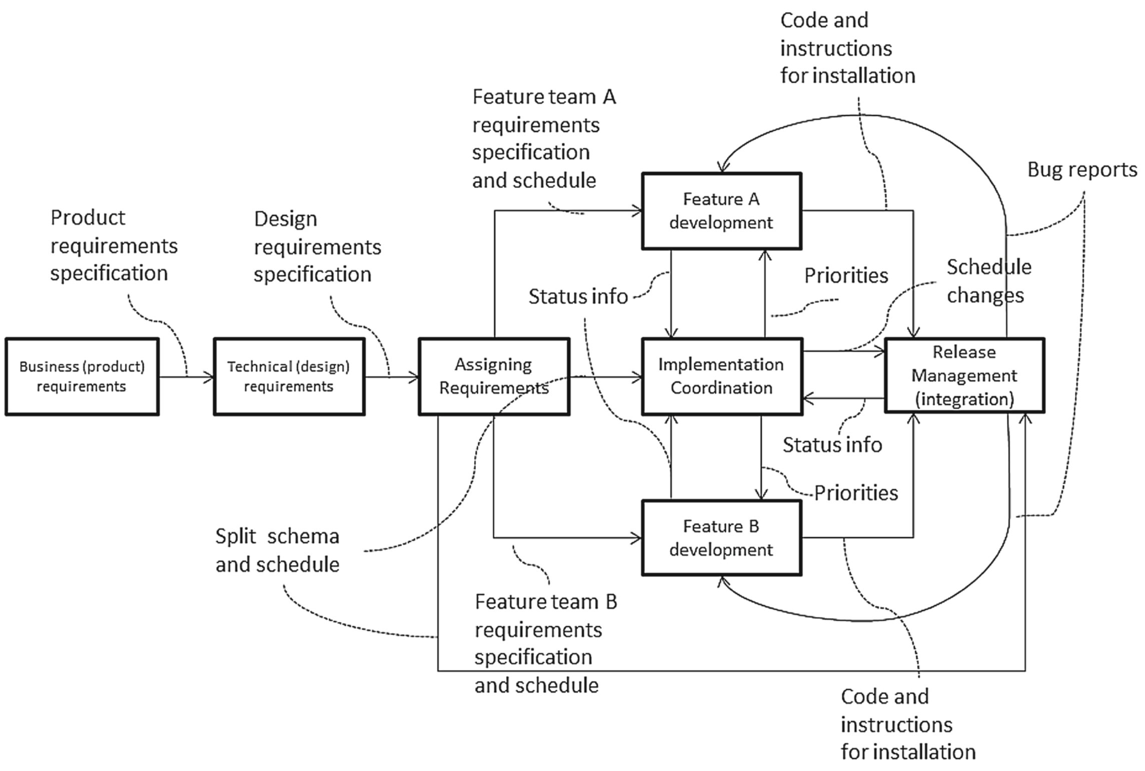
Code and instructions for installation
Feature team A requirements specification and schedule
Bug reports
Feature A development
Product requirements specification
Design requirements specification
Priorities
Schedule changes
Status info
Business (product) requirements
Technical (design) requirements
Assigning Requirements
Implementation Coordination
Release Management (integration)
Status info
Priorities
Split schema and schedule
Feature B development
Feature team B requirements specification and schedule
Code and instructions for installation

Fig. 11 Software development at a large ICT provider—input/output relationships

demonstrated in a meeting that besides the authors and the MS student included two teachers from our department that were totally unaware of the project. The goal of presentation was to get answers on three questions: (a) Whether the model captured all details needed for building a system, (b) whether the prototype gave good understanding of how the system built according to it would work, and (c) whether they were prepared to use this system, if it were built. The answers for questions (b) and (c) from the two teachers that did not participate in the project were positive, namely the prototype gave good understanding of how the system built according to it would work, and they were prepared to use this system, if it were built. The discussion around the first question reveals some concerns, e.g., on the needs for integrating with other systems already in place.

The modeling and prototyping project discussed above represent only partial evaluation of our approach. The real evaluation would require building a system and introducing it in practice. Making a decision on this issue is outside our control, and for now, we do not plan to conduct any further research that concerns this particular case.

Returning back to the model in Fig. 10 and "Appendix 10", it has been drawn by the authors in one brainstorming session. There was no need to engage other people, as the authors were familiar with the process through their own experience as university teachers. It took about two hours to create a step-relationship model of this process along the guidelines suggested in "Appendix 11".

Summarizing our experience in this test case, we can conclude that:

- for a relatively simple business process, it does not take much time to create a step-relationship process model following the guidelines in "Appendix 11" and then derive requirements on a BPM tool according to the guidelines from Section 7;
- a step-relationship model could be useful, if not for selecting a BPM tool, then for selecting a modeling technique appropriate for building a detailed model of the process in question.

8.2 Software development at a large ICT provider

A more complex test that concerned step-relationship modeling was completed at a large ICT provider who recently went through reengineering of their software development process. The reengineering concerned transforming this process from a traditional phase-based development with local software development teams, to a process of working in an iterative manner using the Scrum project management methodology and employing geographically distributed teams that have cultural differences and often work in different time zones.

The new software development process employed a number of tools for managing requirements and test cases, tracking bugs, and problem reports among others. Still, the project

Table 13 Parallel execution matrix for test case 2

	BR	TR	AR	FD-A	FD-B	IC	RM
Business requirements (BR)							
Technical requirements (TR)							
Assigning requirements (AR)							
Feature A development (FD-A)					x	x	x
Feature B development (FD-B)				x		x	x
Implementation coordination (IC)				x	x		x
Release management (RM)				x	x	x	

management felt that there were some problems in the new settings that warranted analysis of the suitability of the tools employed. The decision was made to start a small-scale project to analyze the situation and suggest improvement in both the tools employed and the process organization. It was also decided to use step-relationship modeling in the project.

In contrast to the case in Sect. 8.1, the software development process was quite complex and included many participants. Nevertheless, in the end, it was possible to identify a relatively small number of steps and then define relationships between them. The steps and the major part of input/output relationships between them are presented in graphical form in Fig. 11 (for more details see [13]). As can be seen in Fig. 11, we succeeded in reducing the number of steps to seven for this complex process. In the actual case, the number of feature development steps was more than two. However, the model in Fig. 11, with only two *Feature X development* steps (A and B), represents the process quite well, because adding more feature development steps does not add structural complexity, only quantitative complexity. Most of interactions happen between *Feature X development*, *Implementation coordination* , and *Release management*. The parallel execution matrix for this process is presented in Table 13. This matrix shows that steps *Business requirements*,*Technical requirements*, and *Assigning requirements* are executed in sequential fashion, while *Feature X development*, *Implementation coordination*, and *Release management* run in parallel. As these last steps also have tight input/output coupling, the parallell dependencies matrix has a considerable number of crosses.

All steps have teams. The teams for the first two steps do not intersect with each other and with the teams of the other steps. *Assigning requirements* intersect with *Implementation coordination* and *Feature X development*. *Implementation coordination* intersects with *Feature X development* and *Release management*. Different *Feature X development* teams neither intersect between themselves, nor do they intersect with *Release Management* team.

On the whole, coordinating *Feature X development*s and *Release Management* requires a lot of efforts from the *Implementation coordination* team and does not work smoothly.

In the project, we successfully built a full step-relationship model and derived requirements on tool support for the software development process. As this process already employed a number of tools, completely substituting them was out of the question. The model and requirements were used for analysis of suitability of the tools already employed. For this end, we created a new matrix in which cells showed which tools were used to support (a) relationships between the steps (nondiagonal cells) and (b) step teams work (diagonal cells).

Through analysis of the tools, we found a lack of support for the weak dependencies between *IC* (Implementation Coordination) on one hand and *FD*s (feature development teams) and *RM* (Release Management) on the other hand. To improve support for the weak dependencies, a tool that would provide an integrated picture of the teams' activities seemed necessary. Such a tool could analyze activities around the different requirements assigned to a team, e.g., code compilations, or bug reports, and give warnings on suspicious or too much activity, or lack of such activity.

The project of building a model and giving recommendations was conducted by a team consisting of three members, which included the first author of this paper. The other participants were a MS student from Stockholm University and a researcher from the ICT organization who coordinated the project. The project consisted of the following steps:

1. Identify all units engaged in the software development process;
2. Interview representatives of each unit;
3. Create a hypothesis—draw a step-relationship model based on the material from interviews;
4. Verify and refine the hypothesis by presenting it to the interviewed representatives, and thus developing the final version of the model;
5. Derive requirements on the tool support from the model and compare them with the capabilities provided by the existing tools;
6. Present the model and findings to the representatives of the units who participated in the project and discuss alternatives for continuing the work.

The bulk of the work in the project was completed by the MS student and researcher from the ICT provider. The role of the first author was limited to scientific supervision and providing assistance in creating a hypothesis and formulating findings. A more detailed description of the project was published in [13] and is included in the student's MS thesis.

From the discussions with participants of the software development process (Step 6 above), it became clear that the step-relationship model produced in the project had value in addition to the assessment of the existing tools and suggestions for adding new tools. The model represented the first concise description of the software development process that emerged after accepting Scrum and the geographically distributed teams. Therefore, the model can be used for presenting the process to all its participants, so that they can fully grasp the holistic picture of software development and their role in it.

Based on the results of this limited-scope project, it was decided to start a new feasibility study. The study would concentrate on both (a) tool support for weak dependencies, and (b) using step-relationship modeling for planning and following up all projects inside the company.

Summarizing our experience in this test case, we can conclude that:

- It is possible to develop a concise step-relationship model even for a relatively complex process though it takes time and effort to build the model;
- A step-relationship model can be useful for the assessment of tools already employed for supporting the process, and for giving a holistic view of the process to its participants in order to enable them to make decisions not only from their local point of view, but for the best of the entire process.

8.3 Lessons learned

The cases presented in the previous sub-sections did not include the usage of step-relationship modeling for the sake of selecting a BPM tool, which was the problem that initiated the development of a new modeling technique. We are still looking for an opportunity to employ our technique for this purpose. However, lessons learned in the completed cases can be used for the continuing development of the new modeling technique; in particular, the following lessons could be useful for further research and development.

1. Step-relationship modeling is useful for the tasks "adjacent" to the one of selecting BPM tools, e.g., for assessing already employed tools to discover gaps and misalignment, as was done in the second test case, or for choosing a modeling technique for detailed analysis of the process, as was done in the first case. In addition, the technique can be useful for completely different tasks, as was discovered in the second test case. More particular, it indicated its usefulness for increasing the level of understanding of a complex process by its participants. There may be other areas where step-relationship modeling could be useful, which warrants an additional study.
2. A step-relationship model, when it has been built, is understandable for the process participants, even when they have no formal training in business process modeling, which is one of the lessons from our second case.
3. While the matrix form of presenting step relationships is very convenient for formal analysis and matrices merging, it is not particularly good for visualizing the issues in derived matrices that need special attention. It was revealed that the graphical form of the type of Fig. 11 is much better for brainstorming discussions. A better way for visualizing derived matrices and special issues revealed in them needs to be found in order to increase the usefulness of the model.
4. The key feature of the step-relationships model is derived matrices, especially the parallel dependencies matrix (see Sect. 5.5) and the merger of the weak dependencies matrix and teams matrix (see Sect. 5.8). Analysis of these matrices shows the issues in the process that require special consideration, which helped us to identify areas of improvements in the second case. This warrants looking into other possible combinations of basic matrices to see whether some of them could have meaning and be useful.
5. Building derived matrices manually takes time, and this may lead to human-introduced errors. This work needs to be automated, which is not a particularly difficult task.
6. So far, in all cases where step-relationship modeling has been used, at least one of the authors has participated. However, the main bulk of modeling work in the second case was done by two other members of the project team, one of which was an MS student. In addition, we had one smaller project where two students, in the frame of their BS thesis, built a step-relationship model of a catering process at a small restaurant. Based on our experience, we believe that the skills of step-relationship modeling are transferable, in particular, to professional business consultants that already have experience in the analysis of complex business situations. However, to facilitate the transfer, there is a need to develop practically oriented manuals that include instructions of the type presented in "Appendix 11", alongside examples of their application. In addition, some kind of tool-support for filling matrices would make the knowledge- and skill-transfer easier.

8.4 Promoting adoption

While continuing to look for new practical cases where the suggested solution as a whole, or the step-relationship model can be tested, we accept that the researchers, themselves, cannot create enough cases to prove the validity of their finding. The only way to obtain such a proof is by promoting the

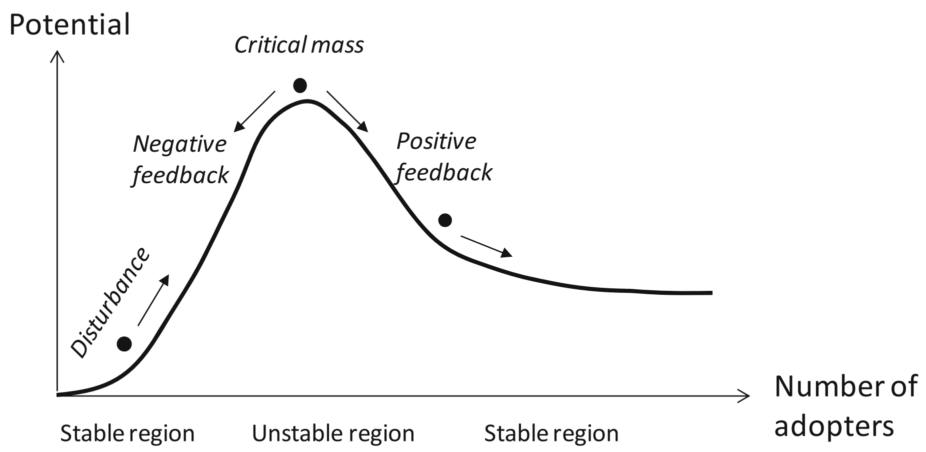

Fig. 12 Adapted from [32]

adoption of our solution and modeling technique so that others can provide enough cases for a rigorous investigation. To conduct the promotion effectively, we need to identify the most promising recipients for adoption.

According to the classical works on the adoption and diffusion of technology [38], successful adoption requires reaching a critical mass of adopters. To ensure that a critical mass is reached in the most effective way, efforts need to be concentrated where the negative feedback can be minimized, and the positive feedback maximized (see Fig. 12). BPM tool vendors may not be considered as an appropriate target for early adoption. Their marketing efforts are directed at promoting their existing tools; thus, a method that reveals that their tools might not be appropriate in some particular cases could be easily rejected. A more appropriate target group would be business consultants and business analysts that are not attached to particular IT or to BPM vendors or solutions. For this target group, new methods that permit the completion of certain tasks with better precision or results would create a competitive advantage. In addition, it will allow them to bring something new to their customers.

Attracting this group, however, requires the completion of a number of tasks that will clearly demonstrate the advantages of the new method:

1. Clearly formulating the areas of applicability in the form of a list of practical problems that the suggested method and new process modeling technique are aimed to solve (see lesson learned #1 in the previous section).
2. The problems in such a list should be formulated in business rather than technical terms. Based on our already existing experience, the list could include the following items: "recommending a BPM tool suitable for a particular process"; "assessing alignment between business and IT"; and "team-building in large software development processes that uses Scrum methodologies." The list will be increased when more test cases have been completed that include step-relationship process modeling.
3. Providing means for visualization of results and IT support that makes it easier to complete a project and demonstrate the results (see lessons learned #3 and #5 in the previous section).
4. Providing descriptions of test cases that demonstrate the results achieved when using the suggested method and modeling technique.
5. Task 4 should be completed via easily available for practitioners' outlets, e.g., Internet, journals directed at practitioners, and business-oriented conferences.
6. Providing manuals with clear description and examples on how the method and modeling technique can be employed in practice (see lesson learned #6 in the previous section).

The above tasks for dissemination of our research results and promoting the adoption of our method and modeling technique are included in our long-term plans. Some of these tasks, for example, developing methods of visualization for step-relationship matrices are planned for the near future.

9 Discussion and plans for the future

9.1 Contribution and potential impact

The practical applications of the solution presented in this paper, and in particular, the new process modeling technique, have been discussed in Sect. 8. Section 8 also includes a strategic plan for promoting the adoption of our suggestions by industry. This section presents a summary of what we believe are the main contributions of this paper in the field of research and what impact they may have on it.

The steadily growing number of publications related to the requirements for BPM tools, such as [21], and for BPS systems in general, such as [6], shows that the research community is aware of the importance of this issue. Our contribution to this awareness is that we have highlighted the need for *deriving requirements from the nature of the process to be supported*, not only from the general issues related to all processes. To the best of our knowledge, this problem has not been clearly formulated in the research literature, although there is tacit awareness of its existence. We believe that our work may inspire other researchers to look for solutions for the problem formulated.

In addition to formulating the problem, this paper outlines a particular solution for it. The solution is based on the idea of building a high-level model of the process with just enough of details to produce a list of requirements on a BPM tool or BPS system. Such a solution is economically advantageous, as it requires less effort than building a detailed business process model. According to our experience, for a relatively simple process, such a model can be built during one facilitating workshop (see Sect. 8.1). For a more complex process, it takes more time, but largely because it requires interviewing many people to understand the whole picture (see Sect. 8.2). Using a step-relationship model also diminishes the risk that a particular choice of modeling technique for producing a detailed model creates a bias in the process of requirements elicitation. Because our solution is not fully developed, we hope that the publication of this paper will encourage others to join our efforts in developing this solution further, and/or in testing it.

The central component of our solution is a new process modeling technique—step-relationship modeling. The main characteristics of this technique are its use of multiple aspects and high level of consideration. The model takes into account several well-known aspects of business processes, such as input–output flow as it is carried out, for example, in IDEF0 [22]. In addition, it formalizes less-known aspects, such as informal information exchange between process participants—in the form of the so-called weak relationships. High level refers to the fact that the level of detail depicted in the model is limited. For example, we do not consider which roles are engaged in each step, but only whether the steps have teams, and whether they intersect or coincide. As far as we know, there is no modeling technique that considers our particular combination of aspects, while keeping the description on a high level. We hope that our work will encourage others to conduct research in multi-aspects and high-level business process modeling.

As has already been summarized in lesson learned #4 (see Sect. 8.3), the most interesting observations on the nature of the process at hand are obtained from an analysis of the derived matrices. Normally, a derived matrix is produced, by merging two or several other matrices, thus revealing the properties that arise from the interaction of several aspects. As far as we know, such interactions are not well researched. Thus, our approach may increase the interest in the analysis of the interaction of various aspects by providing the means to carry it out.

As has already been mentioned in lesson learned #1 (see Sect. 8.3), the step-relationship model appears to be useful[15] in situations other than the one for which it was invented, such as assessment of tools already employed for supporting a business process, or giving a holistic view of the process to its participants. The latter is due to the fact that the model is understandable even for people without formal BMP training (see lesson learned #2 in Sect. 8.3). There might be other fields of application that could be revealed when (and if) practice and research adopt the new modeling technique.

While formulating a problem and discussing a solution, this paper demonstrates the practical usage of DSR. In this respect, the paper can be viewed as a test case for the DSR model suggested in [12]. We found the two-worlds model, described in Sect. 3, to be useful for planning our work, including the planning of the continuation of the research. While working on the second test case (Sect. 8.2), we discovered that the modeling technique that we suggested for solving one problem may be useful for other problems. This warrants formulating other hypotheses in the generic situations-problems-solutions space of Fig. 3 with solutions based on employing step-relationship modeling. In respect to DSR, we hope that its usage, as described in this paper, might raise interest in design science in general, and our approach of planning DSR projects, as described in Sect. 3, and in [12].

9.2 Challenges to overcome

The full solution outlined in this paper can be summarized as the process of choosing BPM tools, as presented in Fig. 13. Only the first four steps were given full attention in the previous sections. In this section, we will look at challenges to overcome when dealing with the last step.

The most immediate challenge is connected to the fact that BPM tools vendors do not describe their tools in terms of the capabilities we have listed. Usually, they do not provide information in such terms as information logistics or process flow restrictions. The descriptions the vendors use are in terms of a business domain for which the tool is recommended, or/and in the form of functional specifications. To formalize the last step in Fig. 13, there is a need to introduce a standard for the capabilities of by BPM tools that can be used by the vendors, or design a practical methodology of analysis of BPM tools that produces a list of these capabilities. In addition, just having a list of capabilities provided by a BPM tool may not be enough because there can be dependencies between them, with the result that some capabilities cannot be provided without the others. These dependencies need to be revealed so that choosing a tool that provides one useful capability, but also includes a set of unusable ones that can be avoided.

[15] Which would satisfy [18] requirement on modeling: "Essentially, all models are wrong, but some are useful," p. 424.

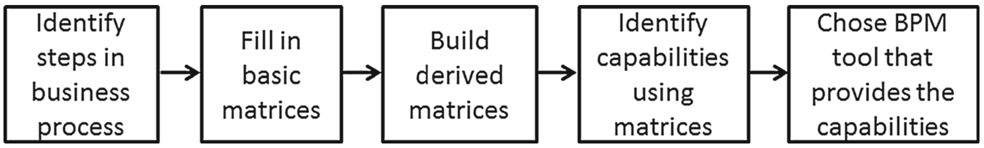
Identify steps in business process
Fill in basic matrices
Build derived matrices
Identify capabilities using matrices
Chose BPM tool that provides the capabilities

Fig. 13 A simplified process for choosing BPM tools

We consider that the idea of a standard forced upon tool vendors has little chance of success. Providing a methodology to analyze a tool and obtain a list of capabilities it supports seems to have a better chance for success. In the end, there could be an established service that analyzes any new tool, or a major modification of an existing tool and publishes a list of the capabilities it supports. The first step that we plan for developing such a methodology is to study the existing BPM tools and the documentation that describes them. The purpose of this study will be to investigate the vocabularies used in different kind of tools and find out how particular capabilities can be expressed in each of the vocabularies.

The next challenge is connected to the fact that it might not be practical to choose a BPM tool that is based only on the characteristics of one process. For example, if the same people participate in several different processes, it would be difficult to introduce, into practice, several BPS systems based on different principles. In such a situation, the choice could be between reaching a compromise when choosing a BPM tool and developing a specialized BPS system. This kind of decision lies outside the scope of this paper, because we believe these decisions need to be made on the case-by-case basis. For such cases, our solution could still be useful, as it can give information on what capabilities are desirable for each particular process, and which BPM tools provide which capabilities. This could help in making an informed decision and reaching a compromise.

9.3 Plans for the future

In Sects. 8.3, 8.4, and 9.2, we have already described what is needed for the further development of our solution and for making it useful for practical purposes. This includes, but is not limited to the following:

1. Developing the means for the visualization of derived matrices. The importance of this activity follows from lesson learned #3 (see Sect. 8.3).
2. Developing a methodology of BPM tools analysis that will help in obtaining a list of capabilities provided by each tool in the market to meet the challenges discusses in Sect. 9.2
3. Finding more test cases to further validate the usefulness of our approach in general and our modeling technique in particular.
4. Developing software to support step-relationships modeling, including automatically producing derived matrices and visual representations of them. The importance of this activity follows from lessons learned #3 and #5 (see Sect. 8.3).
5. Developing practical methodology (manuals) for step-relationship business process modeling. The importance of this activity follows from lesson learned #6 (see Sect. 8.3).

At present, most of our efforts are invested in the first activity on the list. This is being done in the frame of a practical project in cooperation with the same ICT provider, as reported in Sect. 8.2. The project is aimed at building a model of another software engineering process at this provider; this process is similar but not equal to the one described in Sect. 8.2. The goal of the project was to provide a holistic view on the process to be used for its improvement. In this project, we are both extending the model, and investigating the ways of visual presentation of other matrices than the input–output matrix. The extension concerns representation of diversity of step teams with respect to organizational, geographical, and professional affinity of their members, which also need to be represented graphically.

We also started investigation related to the second activity on the list, but for the moment, it goes slowly due to the lack of resources. Pursuing this and the rest of activities on the list will be carried out as resources become available. Besides the research activities, we are negotiating with a group of management consultants on the issue of adoption of our technique.

Acknowledgments The authors are grateful to all members of our team without whose efforts this paper would have never been written. Special thanks to Athanasios Karapantelakis and Nirjal Khadka, who participated in the project aimed at validation of the modeling technique suggested in this project. The authors are also much in debt to the anonymous reviewers whose comments helped us to improve the structure and readability of this paper.

10 Appendix 1: Matrices for the course preparation process model

This appendix contains all matrices for the model of the course preparation process described in Sect. 9.1

10.1 Input–output matrix

Input–output	Plan course	Schedule course	Print course material	Teach and learn	Evaluate
Plan course					
Schedule course	*Activity sequence				
Print course material	*Course material				
Teach and learn	*Working material	*Schedule	Printed course material		
Evaluate	*Evaluation form				

10.2 Transitive input–output matrix

Input–output	Plan course	Schedule course	Print course material	Teach and learn	Evaluate
Plan course					
Schedule course	x				
Print course material	x				
Teach and learn	x	x	x		
Evaluate	x				

10.3 Parallel execution matrix

Input–output	Plan course	Schedule course	Print course material	Teach and learn	Evaluate
Plan course		x	x	x	
Schedule course	x		x	x	
Print course material	x	x		x	
Teach and learn	x	x	x		
Evaluate					

10.4 Parallel dependencies

Input–output	Plan course	Schedule course	Print course material	Teach and learn	Evaluate
Plan course					
Schedule course	x				
Print course material	x				
Teach and learn	x	x	x		
Evaluate					

10.5 Weak dependencies

Input–output	Plan course	Schedule course	Print course material	Teach and learn	Evaluate
Plan course				Modification of teaching and learning activities Modification of material	
Schedule course	Size of groups, elapses between activities, etc.			modification of schedule	
Print course material	Feedback on quality of material				
Teach and learn	Rational for learning and teaching activities				
Evaluate	Rational for learning and teaching activities				

10.6 Teams matrix

Output Input	**Plan course**	**Schedule course**	**Print course material**	**Teach and learn**	**Evaluate**
Plan course	*[Teachers]*				
Schedule course		*[Teachers-Scheduler]*			
Print course material					
Teach and learn				*[Teachers and Students]*	
Evaluate					*[Teachers and Students]*

10.7 Weak dependencies + teams

Output Input	**Plan course**	**Schedule course**	**Print course material**	**Teach and learn**	**Evaluate**
Plan course					
Schedule course	Size of groups, elapses between activities, etc.			Modification of teaching and learning activities. Modification of material	
Print course material	Feedback on quality of material			Modification of schedule	
Teach and learn	Rational for learning and teaching activities				
Evaluate	Rational for learning and teaching activities				

11 Appendix 2: Guidelines for brainstorming process properties

This appendix contains a set of questions that could be used in a facilitating workshop with process participants aimed at building a high-level business process model for the process in question. Based on the answers, one can build all matrices introduced in Sect. 5.

6. Identify major step in the process.

Example software engineering, steps *Requirements*, *Design*, *Coding*, *Test* (See Fig.1)

7. For each step identify the following

a) *Inputs*: things that should be provided to the people working in the step so that they can successfully complete their work. Inputs that needs to be present at the very beginning when work starts are marked with asterisk (*)
b) *Outputs*: the result expected in this step
c) *Size of the step*: team especially whether only one person is engaged in the step or there are more people engaged

Example *Requirements*

a) *Inputs*: None
b) *Outputs*: Requirements specifications; Test specifications
c) *Size of the team*: more than one person

Example *Design*

a) *Inputs*: Requirements specifications*
b) *Outputs*: Design Specifications
c) *Size of the team*: more than one person

8. For each pair of steps A and B determine the following

a) *I/O A to B*: Outputs from A that serve as inputs for B
b) *I/O B to A*: Outputs from B that serve as inputs for A
c) *Parallel execution*?: Whether A and B are allowed to run in parallel
d) *Team Intersection?*: Not intersect, Intersect, Coincide (are the same)
e) *Weak dependencies A to B*: Whether B may require some extra information from A, aside of possible inputs to B that are outputs from A
f) *Weak dependencies B to A*: Whether A may require some extra information from B, aside of possible inputs to A that are outputs from B

Example: A = *Requirements*, B = *Design*

a) *I/O A to B*: Requirements specifications
b) *I/O B to A*: None
c) *Parallel execution*?: Yes
d) *Team Intersection?*: Intersect
e) *Weak dependencies A to B*: Yes, e.g. explanations on the importance of some requirements
f) *Weak dependencies B to A*: No

Example A = *Design*, B = *Requirements*

a) *I/O A to B*: None
b) *I/O B to A*: None
c) *Parallel execution*?: No
d) *Team Intersection?*: Not intersect
e) *Weak dependencies A to B*: No
f) *Weak dependencies B to A*: No

References

1. ActionFlow: ActionFlow web site, available (2012–10-01) at: http://www.actionflow.com (2012)
2. Agile manifesto: Manifesto for Agile Software Development, available (2012–10-01) at: http://agilemanifesto.org/ (2001)
3. Alter, S.: Work system theory: overview of core concepts, extensions, and challenges for the future. JAIS **14**(2), 72–121 (2013)
4. Anderson, J., Donnellan, B., Hevner, A.R.: Exploring the Relationship between Design Science Research and Innovation: A Case Study of Innovation at Chevron, Communications in Computer and Information Science. Springer, Berlin (2011)
5. Andersson, T., Bider, I., Svensson, R.: Aligning people to business processes experience report. Softw. Process Improv. Pract. **10**(4), 403–413 (2005)
6. Anttila, J., Jussila, K.: An advanced insight into managing business processes in practice. Total Qual. Manag. Bus. Excell. **24**(8), 918–932 (2013)
7. Appian: Appian cloud BPM, available (2012–10-01) at:http://www.appian.com/bpm-software/cloudbpm.jsp (2012)
8. Baresi, L., Casati, F., Castano, S., Fugini, M.G., Mirbel, I., Pernici, B.: Workflow design methodology. In: Grefen, P., Pernici, B., Sanchez, G. (eds.) Database Support for Workflow Management: The WIDE Project, pp. 47–94, Kluwer (1999)
9. Baskerville, R.L., Pries-Heje, J., Venable, J.: Soft Design Science Methodology, DERIST 2009, ACM, pp. 1–11 (2009)
10. Bider, I., Bellinger, G., Perjons, E.: Modeling an agile enterprise: reconciling systems and process thinking. In: Johannesson, P., Krogstie, J., Opdahl, A. (eds.) The Practice of Enterprise Modelling, IFIP, LNBIP, vol. 92, pp. 238–252, Springer, Berlin (2011a)
11. Bider, I., Johannesson, P., Schmidt, R.: Experiences of using different communication styles in business process support systems with the shared spaces architecture, Proceedings of CAiSE 2011, LNCS, Vol. 6741, pp. 299–313, Springer, Berlin, (2011b)
12. Bider, I., Johannesson, P., Perjons, E.: Design science research as movement between individual and generic situation-problem-solution spaces. In: Baskerville, R., De Marco, M., Spagnoletti, P. (eds.) Organizational Systems. An Interdisciplinary Discourse, pp. 35–61. Springer, Berlin (2013a)
13. Bider, I., Karapantelakis, A., Khadka, N.: Building a high-level process model for soliciting requirements on software tools to support software development: experience report. PoEM 2013 (Short Papers). CEUR **1023**, 70–82 (2013)
14. Bider I., Perjons E., Johannesson P.: In Search of the Holy Grail: Integrating social Software with BPM. Experience Report. LNBIP, No 50, pp 1–13, Springer, Berlin (2010)
15. Bider, I., Perjons, E.: Evaluating adequacy of business process modeling approaches. Handb. Res. Complex Dyn. Process Manag. Techn. Adapt. Turbul. Environ., IGI, 79–102 (2009)
16. Bider, I., Perjons E.: Preparing for the Era of Cloud Computing: Towards a Framework for Selecting Business Process Support Services. Enterprise, Business-Process and Information Systems Modeling, LNBIP, No 113, pp. 16–30, Springer, Berlin (2012)
17. Bider, I., Perjons, E., Riaz Dar, Z.: Using Data-Centric Business Process Modeling for Discovering Requirements for Busi-

ness Process Support Systems: Experience Report. Enterprise, Business-Process and Information Systems Modeling, LNBIP 147, pp. 63–77, Springer, Berlin (2013c)
18. Box, G.E.P., Draper, N.R.: Empirical Model Building and Response Surfaces. Wiley, New York (1987)
19. BPMN: Business Process Model and Notation website, Object Management Group, available (2012–10-01) at: http://www.bpmn.org/ (2012)
20. Cohn, D., Hull, R.: Business artifacts: a data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. **32**(3), 3–9 (2009)
21. Duarte Filho, N. F., Costa, N. P.: A set of requirements for business process management suite (BPMS). In: Advances in Information Systems and Technologies, pp. 487–496, Springer, Berlin (2013)
22. FIPS: Standard for integration definition for function modeling (IDEF0). National Institute of Standards and Technology (NIST). FIPS publication, p. 183 (1993)
23. Gilbert, P.: The next decade of BPM. Keynote presentation at BPM 2010 conference, available (2012–10-01) at: http://www.bpm2010.org/wp-content/uploads/2009/08/2010-09-14-Gilbert-BPM-2010-Keynote.pdf (2010)
24. Giere, R.: Scientific Perspectivism. University of Chicago Press, Chicago (2010)
25. Hammer, M., Champy, J.: Reengineering the Corporation: A manifesto for Business Revolution, Harper Business, New York (1993)
26. Hant, V.D.: Process Mapping: How to Reengineer Your Business Processes. Wiley, New York (1996)
27. Hevner, A.R., March, S.T., Park, J.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
28. iPB: iPB Reference Manual, available (2012–10-01) at: http://docs.ibissoft.se/node/3 (2012)
29. Khan, R.: Evaluating BPM software. Bus. Integr. J. October issue (2003)
30. Khomyakov M., Bider, I.: Achieving Workflow Flexibility through Taming the Chaos, OOIS 2000–6th international conference on object oriented information systems, pp. 85–92, Springer, Berlin (2000)
31. Kleiner, N.: The focus of requirements engineering in workflow application development. CEUR Workshops Proc. **75**, 372–377 (2003)
32. Link, E.: Diffusion Dynamics and Pricing of Innovations. Lund University Press, Lund (1997)
33. Müller, D., Reichert, M., Herbst, J.: Data-Driven Modeling and Coordination of Large Process Structures. In OTM2007, PartI, LNCS 4803. pp. 131–49, Springer, Berlin (2007)
34. Neuman, W.L.: Social Research Methods. Qualitative and Quantitative Approaches, Sixth Edition, Peason (2006)
35. Österle, H., Becker, J., Frank, U., Hess, T., Karagiannis, D., Krcmar, H., Loos, P., Mertens, P., Oberweis, A., Sinz, E.J.: Memorandum on design-oriented information systems research. Eur. J. Inf. Syst. **20**(1), 7–10 (2010)
36. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inf. Syst. **24**(3), 45–78 (2007)
37. Projectplace: Project Place website available (2012–10-01) at: http://www.projectplace.com (2012)
38. Rogers, E.M.: Diffusion of Innovations. Free Press, New York (1962)
39. SalesForce: SaleForce website, available (2012–10-01) at: http://www.salesforce.com (2012)
40. Sein, M.K., Henfridsson, O., Purao, S., Rossi, M., Lindgren, R.: Action design research. MIS Q. **35**(1), 37–56 (2011)
41. SpringCM: Content cloud services, available (2012-10-01) at: https://www.springcm.com/products/springcm-content-cloud-services (2012)
42. Swenson, K.D. (ed.): Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done. Meghan-Kiffer Press, Tampa (2010)
43. Van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. **53**(2), 129–162 (2005)
44. Van der Aalst, W.M.P., Van, Hee K.M.: Workflow Management: Models, Methods and Systems. MIT Press, Cambridge (2002)
45. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2007)
46. Wieringa, R.J.: Design science as nested problem solving. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, ACM, Philadelphia, pp. 1–12 (2009)
47. Wohed, P., Andersson, B., Johannesson, P.: Open source workflow systems, Mod. Bus. Process Automa. 401–434 (2010)

Ilia Bider Docent at the department of Computer and System Sciences of Stockholm University and co-founder of IbisSoft AB, Sweden. Dr. Ilia Bider is Software Engineer, Business Analyst, IS-researcher and Teacher with long experience in several IT-related fields. He has MS in Electronic Engineering and PhD in Computer and System Sciences, and combined experience of over 30 years of research (in the fields of IS, SE, DB, and computational linguistics), and practical work (business analysis, and software design, coding, sales, and marketing) in five countries (Norway, Russia, Sweden, UK, and USA). Dr. Bider has published over 50 research papers as well as a considerable number of articles for practitioners. His main specialty is finding research topics in business practice, and testing research results in business practice. Dr. Bider is an inventor of the state-oriented approach to business process modeling and control that is based on the application of the conceptual ideas of the Mathematical system theory to the realm of business processes. This approach has been successfully tested in business analysis and software application development practice of IbisSoft and its partners. Dr. Bider puts a lot of effort in bridging the gap between the academics and practitioners.

Erik Perjons PhD, holds a position as senior lecturer at the Department of Computer and Systems Sciences (DSV), Stockholm University. His research and teaching interest include areas such as enterprise modeling, service-oriented architecture, business process management, knowledge management, and decision support systems. A main research focus has been to design methods based on modeling techniques and scientific approaches for analyzing practical problems in organizations, and designing business and IT solutions addressing these problems. He has published over 60 publications in international journals and conferences, and participated in several domestic and international research projects in domains such as health care, e-government, and telecom.