Business Process Model Driven Approach for Automatic Use Case Model Generation

Salam Turkman and Adel Taweel(✉)

Computer Science, Birzeit University, Birzeit, Palestine
salma.turk@gmail.com, ataweel@birzeit.edu

Abstract. Requirement elicitation is an essential step for establishing software requirements. They define the outcomes upon which software functionality is produced. However, several studies have shown majority of errors found in software functionality are directly linked to requirement elicitation. To address, this paper proposes a structured approach to derive system requirements automatically using business process models. It employs a systematic mechanism to improve business process models and transformation method to generate requirement models. It employs 26 defined heuristics rules that maps and controls transformation. The proposed approach is evaluated using seven case studies. Results show the viability to generate software requirements from business process models, and the automatic generation of rich UML-based use case diagram. The proposed approach achieves more precise and valid requirement specifications and was able to generate additional valid use case model features compared to other competing approaches.

Keywords: Requirement engineering · Business process modelling · Use case model

1 Introduction

Effective requirement elicitation is an essential process in developing software applications [19]. It necessitates using appropriate requirements elicitation methods for developing successful software projects. They play an important role in determining whether a project delivers the desired business value and meet management constraints, in terms of time and budget [20]. Many large projects fail due to errors in requirement elicitation, with some studies reporting such errors can be very difficult to discover and very expensive to fix [10]. Several alternative promising approaches based on business process models have been proposed to address these issues [21–23]. They identify business processes as essential element to determine software requirements and user needs and demands from the software applications that provide them [22].

Several business process-based approaches have additionally proposed enabling automatic generation of requirement specifications from underlying organization business models, overcoming errors that often result from traditional manual-based requirement elicitation methods. However, these approaches require significant manual intervention. A previous paper proposed a general approach that achieves better transformation of software specifications from business models and reduces manual intervention

B. Shishkov (Ed.): BMSD 2020, LNBIP 391, pp. 123–136, 2020.
https://doi.org/10.1007/978-3-030-52306-0_8

significantly [29]. This paper proposes an approach (named BMSpec) that uses business process models to derive more accurate software requirements, which in majority can be generated automatically. It proposes a new business model-driven approach for the automatic derivation of UML-based requirement specifications from existing business process models. A key advantage of business process modelling that it enables the engagement of end users in requirement elicitation. The proposed approach, equally enables the engagement of end users even longer, additionally in the business process model re-design which helps to produce better engineered business requirements and solution. The proposed approach developed a systematic method that encompasses a set of heuristic rules that transforms an existing business process model into a BMPN-conforming business model inclusive of prospective "To-Be" processes [21]. Introducing a well-defined business process "To-Be" modelling step for generating requirement specifications results into a more successful software product due to the additional engagement of end users in this business re-engineering stage, unlike traditional approaches that usually include only initial engagement of users. The "To-Be" business process model is then automatically transformed to a Use-Case Model, inclusive of both the UML-based Use-Case diagram and the detailed Use Case descriptions.

The proposed approach has been evaluated on seven case studies. Results show accurate transformation compared with other gold-standard based traditional approaches. They also show the dependency, of the transformation, on the input business process model, whereas the more details it includes, the higher the efficiency of the transformation. The generated requirement specifications are represented as Use Case models that include detailed use case descriptions and associations between use cases within minimal manual intervention.

2 Related Work

Derivation of software specification from business process models has been studied by several researchers [1–4, 13, 22]. Some of the approaches proposed an algorithm to automatically transform business model into functional requirements, as a use case diagram [1]. It works by creating meta-models for both the use case diagram and business process model then compares definitions in the two meta-models. It then attempts to map between concepts from across the two models. However, the total error percentage in the generated use case diagram was relatively high at 40%. Another approach used RADs (Role Activity Diagrams) to model business processes to attempt finding associations for use case diagram derivation [2]. However, the notions of a "Role" and an "Actor" in RADs were not clear enough and incompatible with UML, which resulted into a not well-formed use case diagram.

The notion of "automated activity" in an improved RAD model was suggested to map to the notion of "action or function" in the use case diagram to improve the derivation [3]. This approach, however, has shown lack of process visibility with focus on use cases only with no notation of associations between them. A manual transformation to obtain use case model based on business process models was proposed [13]. Although it provided the first approach that attempted to generate use case description from business process models, but it did not focus on generating the use case diagram. A set of rules, predefined from natural language sentences, were used to generate the use case diagram manually from BPMN elements, the resultant use case diagram however did not cover the association between use cases such as extend, include, invoke and precede. A similar manual approach, using business-oriented to requirement elicitation model (BORE), for deriving system requirements from business process models was proposed to integrate requirements engineering and business process engineering [4]. This may be particularly useful when system requirements are in need of early discovery.

The use of DEMO (dynamic essential modeling of organizations) was investigated, as an alternative approach, to find most suitable methods to identify correct use cases [22]. The suggestion is to use a combination of several requirement generation approaches together to enhance use case derivation. Although, the above demonstrate the potential of automatic generation of software specifications from business models, the need is to automate the derivation consistently using standardized notations. Hence the aim of this paper, it proposes a systematic approach based on well-defined heuristic rules on standardized business process BPMN notation.

3 Proposed Approach

The validity of the underlying business process model is the key to represent the process of the organization business processes. A business process model function does not only represent the overall business process, but also can be used for decision-making within an organization. To do so, however, this requires the business model to be sufficiently detailed and consistently represented with valid modeling notations. Therefore, if well defined, such model can be used to generate requirements for the software needed to support business processes.

Thus, the proposed approach developed two new methods to ensure consistent derivation of use cases from business models:

1) TO-BE model preparation: a structured and systematic TO-BE business model preparation method that ensures the production of a valid model. It takes an existing AS-IS model as input and produces a well-defined valid TO-BE model as output. The validity of the TO-BE model is ensured by confirming the consistency of the constraints, the correctness of the representation and modeling notations. The success of the final result depends mainly on this step, because end users as well participate in developing the TO-BE model, and thus the strength of this approach is in enabling the engagement of end users in the design stage and not just in early stage of requirement modelling.
2) USE-CASE transformation: It takes the TO-BE model as input and produces a USE-CASE model as output. This method developed a set of heuristics rules that are employed to derive and transform the TO-BE model into a USE-CASE model.

In the TO-BE model preparation method, to address the input AS-IS model, the proposed approach employed requirement and business model engineering integration from [4]. This step requires manual processing and the required effort depends on the status of the existing AS-IS business process model, in terms of process details, consistency of application of business constraints and of modeling notations. For the USE-CASE transformation, it developed an algorithm that transforms input TO-BE BMPN XML objects into software specifications represented as UML-based USE-CASE XML objects. BMSpec's transformation algorithm employs a developed set of heuristics rules, see Sect. 2.1, that defines relations between TO-BE business process representation (objects) and UML-based USE-CASE representation (objects). These methods are described in further details in the following subsections.

3.1 TO-BE Model Preparation

To achieve consistent transformation of business process models to use-case software specifications, models need to adhere to a consistent and valid representation. However, due to the variation of existing business models in terms of representation, modeling and validity and consistency in the use of modeling (BPMN) notations, a manual transformation is adopted. Manual transformation ensures that AS-IS models are consistently transformed into TO-BE models. Understandably this may introduce additional effort, and requires modeling expertise, but does not require software engineering expertise. The required level of effort heavily depends on the status of the AS-IS model. Where the input is just a manual AS-IS model and does not cover the user interaction with the system, it requires reengineering to build a valid TO-BE model. This may include analysis of the purpose and effect of the software information system on the business processes. At the end of this step is to identify clearly the automated tasks that represent user interactions with the system. Therefore, it is important to follow good modeling practices to result into significant business improvement [6]. In business process reengineering, processes need be reengineered while taking full advantage of automation to improve business outputs, which may require changing how the business function [7, 8]. Both process and information flows need to be considered along with how people interact with systems to achieve integrated business functions [8].

For maximum benefit, process reengineering should focus on both system perspective, i.e. to reach a clearly defined use case, and business process perspective, i.e. to reach clearly defined business needs from the information system [9]. In this step, the proposed approach aims to identify user interaction tasks with the software system. These tasks often represent important functions, or use cases, that the use case model must include. In other words, each identified use case specifies a functionality that the system must provide for an actor to achieve a business function or process. The proposed approach assumes AS-IS model is represented using BPMN notations. To guide process reengineering, it defines a set of rules to achieve the TO-BE business process model:

- **Rule 1:** Define automated tasks that are achieved fully by the system without any action from user as service tasks.
- **Rule 2:** Define tasks that represent an action of the user on the system.
- **Rule 3:** Remove manual tasks (<<task>>) that cannot be achieved by the system.
- **Rule 4:** Ensure gateway (<<gateway>>) are appropriately defined and used.
- **Rule 5:** Specify <<events for task>> not only <<start events>> but also <<intermediate events>> and <<end events>>.
- **Rule 6:** Define all participants as <<Roles>> not as <<pool>> or <<lane>>.

- **Rule 7:** Assign appropriate (performer) for each user <<task>>.
- **Rule 8:** Specify required <<data objects>>.
- **Rule 9:** Specify required <<message flow>>.

On the other hand, in cases where the existing AS-IS models are designed with a clear software system purpose and the role of the software system on the business is clearly defined, the transformation to the TO-BE model is straightforward. However, a manual check is needed to confirm validity and consistency of the use of BPMN modeling and notation. This requires to ensure correct use of BPMN notations, for example, for specifying task types, declaring conditions for gateways, events name and types. To conform to BPMN notations, the proposed approach defines the following set of rules to guide this modification:

- **Rule10:** Set task type as service task, if the task is fully executed by the system.
- **Rule 11:** Set notation description correctly obeying BPMN's notations, i.e.:
- **Rule11.1**: set a name for each <<gateway>>.
- **Rule11.2**: Set a name for each <<condition>>.
- **Rule11.3**: Set a name for each <<data object>>.
- **Rule11.4**: Set a name for each <<data store>>.
- **Rule11.5**: Set a name for each <<start event>>.
- **Rule11.6**: Set a name for each <<intermediate event>>.
- **Rule11.7**: Set a name for each <<end event>>.
- **Rule11.8**: Set a name for each <<message flow>>.

3.2 Use Case Diagram Transformation: Heuristic Rules

Once TO-BE model is re-engineered, it is used as input to for the transformation. The proposed approach depends on the TO-BE model to be consistent and valid. Therefore, it developed a set of heuristic rules that conform to the correct and consistent use of BPMN notations. The heuristic rules are then used to identify and generate actors, use cases, and their associations and transform them into a use case diagram. Therefore, the developed heuristic rules are based on consistency and normalization of the BPMN notations.

To achieve, BPMN have been studied as a language, not just in terms of its notations but also their semantic use and meaning. Although published BPMN resources and guidelines were used as a reference, but in many cases notation and their semantic uses are not précised defined. Therefore, to reach a better-defined semantic use of the notations, 70 real-time business models have been studied and analyzed to arrive at consistent transformation and interruptions of BPMN notations and their combinations. This helped reach a better understanding of applied uses of the notations and to develop heuristic rules that caters variable semantic uses. Generally, heuristic rules define semantic mapping between business model BPMN notation and software requirement UML notation. For example, a task or activity represents user interaction with the system. In a use case diagram, a use case presents functionality a user wants to achieve through the software system [12]. Thus, a heuristic rule takes the assumption that each activity can be mapped into a use case.

To enable automatic transformation, a computational algorithm was developed based on the heuristic rules. The algorithm takes, as input, the TO-BE model represented as BPMN XML objects, saved from a BPMN modeling tool, combines and applies the heuristic rules to generate the USE-CASE model. The generated USE-CASE model, is generated as XML objects, formatted and used as input to UML-based modeling tool, to generate a USE-CASE diagram. The following describes the developed heuristics rules.

HRule1: Map each activity to a use case. The use case name is the activity name (Fig. 1).

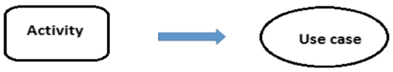

Fig. 1. HRule1.

HRule2: Map each performer to an actor. The actor name is the role (Fig. 2).

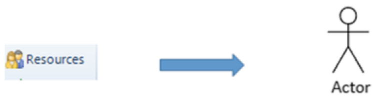

Fig. 2. HRule2.

HRule3: Map each association between an activity and performer to association between actor and use case in use case diagram (Fig. 3).

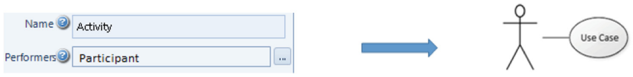

Fig. 3. HRule3.

HRule4: Map sequence flow between two activities to "Precede" association between the corresponding use cases (Fig. 4).

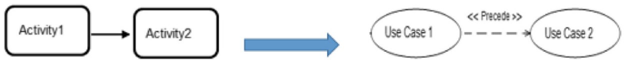

Fig. 4. HRule4.

HRule5: Map exclusive decision gateway between two activities to "extend" association between the corresponding use cases (Fig. 5).

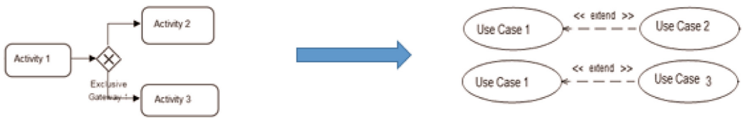

Fig. 5. HRule5.

HRule6: Map data association between activity and input data store to include association between the corresponding use cases (Fig. 6).

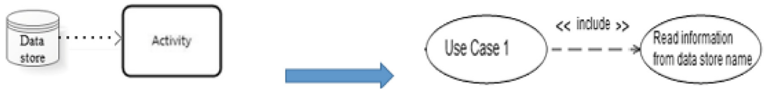

Fig. 6. HRule6.

HRule7: Map data association between activity and output data store to include association between the corresponding use cases as shown in an output data store (Fig. 7).

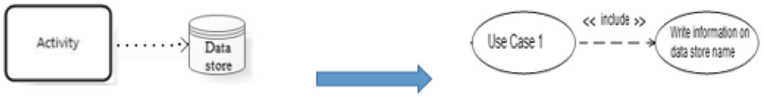

Fig. 7. HRule7.

HRule8: Map data association between activity and input data object to include association between the corresponding use cases as shown in an output data store (Fig. 8).

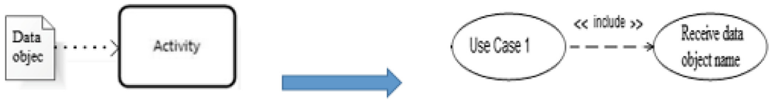

Fig. 8. HRule8.

HRule9: Map data association between activity and output data object to include association between the corresponding use cases as shown in an output data store (Fig. 9).

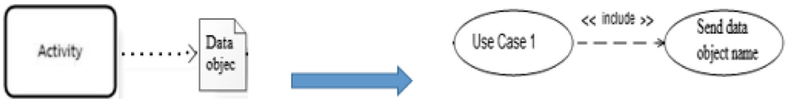

Fig. 9. H Rule9.

HRule10: Map sequence flow between activity and service activity to Invoke association between the corresponding use cases (Fig. 10).

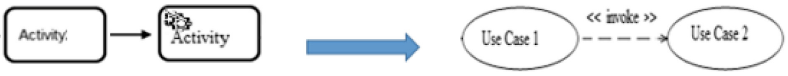

Fig. 10. HRule10.

4 Evaluation and Results

The evaluation methodology aims to evaluate the reproducibility, correctness and validity of the proposed approach's generated use cases, actors and association represented into a use case diagram. Validity examines each of the individually identified or generated use cases, actors and their corresponding associations. Correctness measures the validity of each type of the generated elements and calculates percentage of correctly valid generated elements from total number of elements. In other words, for every input case study and its business process model it should produce or generate a valid use case diagram with correctly identified use cases, actors and associations between them. This is represented in the following equation:

$$\text{Correctness } \% = \text{VEt} - \text{IEt}/\ \text{TEt}$$

Where VE: Number of Valid elements of type t; IE: Number of Invalid elements of type t; TE: Total number of generated elements of type t; t: type of generated elements.

The traditional gold standard testing or model evaluation is used [18]. This is conducted by evaluating the output of traditional requirement elicitation techniques, in which software engineers are employed to build use case diagrams manually, against BMSpec generated use case diagrams, for the same scenarios. Although this is an expensive procedure, yet to ensure validity, the evaluation was done on several different case studies.

Table 1 lists the seven evaluated case studies, including a brief description of each. Results from the evaluation are shown in Tables 2 to 8. 6 of these case studies used traditional requirement engineering elicitation techniques to develop their requirement specification as use case diagrams, and 1 case use used manual transformation from business process model to requirement specifications. As shown, the proposed approach was able to identify extra features, which are not supported in other competing approaches [1–4, 14], such as association between use cases (precede, invoke, include, extend).

Table 2 shows outputs from both the traditional manual approach and the BMSpec automated approach for case study 1 (Nobel Prize). In this case study, 10 use cases, 4 actors and 4 associations were manually identified, which the proposed approach has correctly automatically generated from the corresponding business process model including correct generation of associations between use cases, achieving 100% correctness. For this case study, to achieve, it correctly employed Heuristic Rules: **HRule1, HRule2, HRule3, HRule4, HRule5, HRule6, HRule7, HRule10.**

Table 3 shows outputs from both approaches for use case study 2 (Car hire). In this case study, 3 use cases, 2 actors and 4 associations were manually identified, which the proposed approach has correctly automatically generated from the respective business process model including correct generation of associations between use cases, achieving 100% correctness. For this case study, to achieve it correctly employed Heuristic Rules: **HRule1, HRule2, HRule3, HRule4, HRule7, HRule9.**

Similarly, Tables 4, 5, 6, 7, 8 show outputs of both approaches for remaining case studies respectively, generating their respective use case diagram elements correctly in all cases. Figure 11 shows the output use case diagram compared to the manually generated use case diagram for the X-Road Registration case study.

Table 1. Descriptions of evaluated case studies.

Case study name	Case study brief description
Nobel Prize example	A case study in which a paper work [13] used manual transformation from business process model for the Nobel prize case study to generate a use case model. The authors used a set of rules for the manual transformation. We used the same business process model for the Nobel prize case study used in [13], and the validation was conducted against their derived models
Car-Hire case study	A case study in which a team of four expert master degree students developed a software system for car-Hire Company. The team used a traditional requirement elicitation technique to model system requirements. The output system requirement specifications (SRS) document included a UML use case model and a detailed use case description. The final work of the team has been manually checked and validated
Online Bookshop case study	A case study, in which a team of four expert master degree students developed an online bookshop system. The system enables publishers or book suppliers to setup online shops, and customers to browse and search through the shop and purchase books online. The team used a traditional requirement elicitation technique to model system requirements, and generated an SRS document inclusive of UML use case diagram and use case descriptions, which was manually checked and validated
Three X-road services/case studies	Three case studies, obtained from the Ministry of Telecom and Information Technology (Palestine), each representing a service. These processes represent the daily work of three X-Road services: registration on X-Road service, Consume X-Road service and provide X-Road service. A team of two experts from the ministry used a traditional requirement elicitation technique to derive system requirements for the three case studies. The SRS document included detailed description of the X-road services, UML use case diagram and use case descriptions

Table 2. Nobel Prize example case Study.

	Manual		BMSpec		Correctness	Extra elements
	Qty.	valid	Qty.	valid		<<include>> <<extend>> <<precede>> <<invoke>>
Use cases	10	10	10	10	100%	
Actors	4	4	4	4	100%	

Table 3. Car Hire case study (hire car process) case study.

	Manual		BMSpec		Correctness	Extra elements
	Qty.	valid	Qty.	valid		<<precede>> <<include>>
Use cases	3	3	3	3	100%	
Actors	2	2	2	2	100%	

Table 4. Online Bookshop System (make order process).

	Manual		BMSpec		Correctness	Extra elements
	Qty.	valid	Qty.	valid		<<extend>> <<precede>> <<include>>
Use cases	5	5	5	5	100%	
Actors	3	3	3	3	100%	

Table 5. Extra feature calculator.

	Manual		BMSpec		Correctness	Extra elements
	Qty.	valid	Qty.	valid		
Use cases	2	2	2	2	100%	
Actors	1	1	1	1	100%	

Table 6. Registration on X-Road.

	Manual		BMSpec		Correctness	Extra elements
	Qty.	valid	Qty.	valid		
Use cases	4	4	4	4	100%	
Actors	2	2	2	2	100%	

Table 7. Consume X-Road service.

	Manual		BMSpec		Correctness	Extra elements
	Qty.	valid	Qty.	valid		
Use cases	7	7	7	7	100%	
Actors	3	3	3	3	100%	

Table 8. Provide X-Road service.

	Manual		BMSpec		Correctness	Extra elements
	Qty.	valid	Qty.	valid		
Use cases	5	5	5	5	100%	
Actors	2	2	2	2	100%	

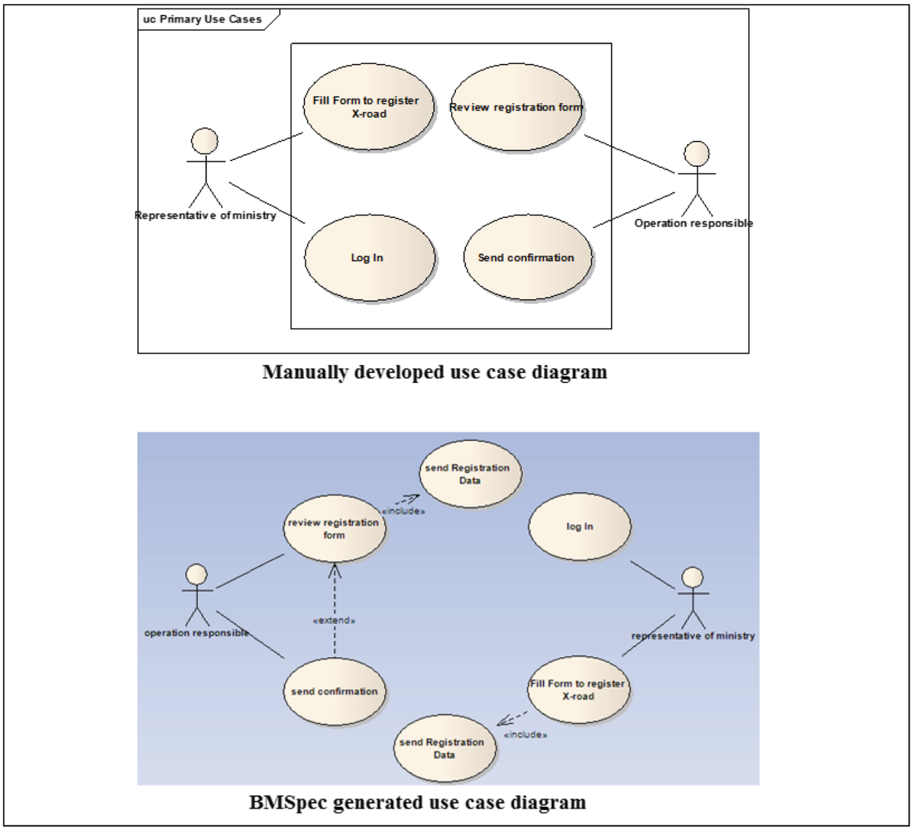

Fig. 11. Use case diagrams for the X-Road Registration service

As shown above, the proposed approach achieved 100% correctness, i.e. identified elements of use case diagrams correctly for all evaluated case studies. Also considering the proposed approach with other competing approaches including [14, 27], it was able to identify and generate additional use case diagram elements and features, including the <<extend>>, <<include>>, <<precede>> and <<invoke >> associations.

5 Conclusion

The paper proposed an approach that shows business models can be used to generate accurate software specifications. The proposed approach employs systematic method that takes a number of systematic steps that processes business process models as input using standardized BPMN notations and produces software specification as UML-based use cases. The approach is evaluated, using gold standard, on seven case studies. Specifications, as UML use case diagrams, of the evaluated case studies were developed using manual traditional requirement engineering techniques and automatically generated using the proposed approach from their respective business process models. Each of the case studies was compared against manually developed traditional requirement engineering techniques. Results show both high generation correctness and efficiency. However, the efficiency of the generation was found to be directly proportional to the level of richness of the input business process model.

While the proposed approach improves the efficiency of the automatic generation of UML-based use case diagrams, it does not however cover or replace the entire requirements engineering stage, nor aims to generate comprehensive requirement specifications. It provides, however, an important step forward to semi-automate the elicitation process through the extraction of as many as possible of requirements from underlying business process models, thus potentially significantly saving development time, reducing requirement misunderstanding errors and improving correct requirements representation using software industry de facto UML.

6 Future Work

One key challenge in requirement modelling, which this approach could be improved to include, is automating requirement traceability, so that any changes in business process model will be reflected automatically in the requirement specifications (e.g. UML use case model, UML activity diagram and UML class diagram).This could solve the problem of volatility that arises in the requirement elicitation phase, and would allow automating the tracking of business growth and changes. This improvement would additionally increase the engagement of end users in the preparing and re-engineering of business models.

References

1. Dijkman, R.M., Joosten, S., Ordina, F.: An algorithm to derive use case diagrams from business process models. In: Proceedings of the 6th International Conference on Software Engineering and Applications, SEA, Anaheim, US (2002)
2. Odeh, M., Richard, K.: Bridging the gap between business models and system models. Inf. Softw. Technol. **45**(15), 1053–1060 (2003)
3. Aburub, F.: Activity-based approach to derive system models from business process models. In: International Conference on Information Society, i-Society. IEEE (2012)
4. Przybylek, A.: A business-oriented approach to requirements elicitation. In: International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE. IEEE (2014)

5. Boehm, B.W.: Software Engineering Economics, vol. 197. Prentice-hall, Englewood Cliffs (1981)
6. Weerakkody, V., Currie, W.: Integrating business process reengineering with information systems development: issues & implications. In: van der Aalst, Wil M.P., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 302–320. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44895-0_21
7. Hammer, M.: Reengineering work: don't automate, obliterate. Harvard Bus. Rev. **68**(4), 104–112 (1990)
8. Kaplan, R.B., Murdock, L.: Rethinking the corporation: core process redesign. Mckinsey Q. **22**, 27–44 (1991)
9. Eriksson, E., Magnus, P.: Business modeling with UML. Wiley, New York (2000). Business Patterns at Work
10. Dijkman, R., Jorg, H., Jana, K., (eds) Business Process Model and Notation. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-39940-9_1195
11. Pressman, R.S.: Software Engineering: A Practitioner's Approach. Palgrave Macmillan, London (2005)
12. OMG: Unified modeling language (OMG UML), version 2.5, Technical report, Object Management Group (2012)
13. Bloch, M., Sven. B., Jürgen, L.: Delivering large-scale IT projects on time, on budget, and on value, McKinsey Quarterly (2012)
14. Cruz, E.F., Machado, R.J., Santos, M.Y.: From business process models to use case models: a systematic approach. In: Aveiro, D., Tribolet, J., Gouveia, D. (eds.) EEWC 2014. LNBIP, vol. 174, pp. 167–181. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06505-2_12
15. Indulska, M., Recker, J., Rosemann, M., Green, P.: Business process modeling: current issues and future challenges. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 501–514. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02144-2_39
16. Rajagopal, P., Lee, R., Ahlswede, T., Chiang, C.C., Karolak, D.: A new approach for software requirements elicitation. In: 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and ACIS International Workshop on Self-Assembling Wireless Network, pp. 32–42. IEEE (2005)
17. Brooks, F.P.: No silver bullet: essence and accidents of software engineering. IEEE Comput. **20**, 10–19 (1987)
18. Eriksson, H.-E., Penker, M.: Business Modeling with UML: Business Patterns at Work. Wiley, Hoboken (2000)
19. Kruchten, P.: The Rational Unified Process – An Introduction. Addison-Wesley, Boston (2000)
20. Davis, A., Dieste, O., Hickey, A., Juristo, N., Moreno, A.M.: Effectiveness of requirements elicitation techniques: empirical results derived from a systematic review. In: 14th IEEE International Requirements Engineering Conference, RE 2006, pp. 179–188. IEEE (2006)
21. Neill, C.J., Laplante, P.A.: Requirements engineering: the state of the practice. IEEE Softw. **20**(6), 40–45 (2003)
22. Berry, D.M., Kamsties, E.: Ambiguity in requirements specification. In: do Prado Leite, J.C.S., Doorn, J.H. (eds) Perspectives on Software Requirements. The Springer International Series in Engineering and Computer Science, vol 753. Springer, Boston (2004). https://doi.org/10.1007/978-1-4615-0465-8_2
23. Kitchenham, B.A., Pickard, L., Linkman, S., Jones, P.: A framework for evaluating a software bidding model. Inf. Softw. Technol. **47**(11), 747–760 (2005)
24. Jalote, P.A.: Concise Introduction to Software Engineering. Springer, London (2008). https://doi.org/10.1007/978-1-84800-302-6

25. Mili, H., Tremblay, G., Jaoude, G.B., Lefebvre, É., Elabed, L., Boussaidi, G.E.: Business process modeling languages: sorting through the alphabet soup. ACM Comput. Surv. (CSUR) **43**(1), 4 (2010)
26. Bider, I.: State-oriented business process modeling: principles, theory and practice (Doctoral dissertation, Data-och systemvetenskap) (2002)
27. Hove, S.E., Anda, B.: Experiences from conducting semi-structured interviews in empirical software engineering research. In: 11th IEEE International Software Metrics Symposium, METRICS 2005, pp. 10. IEEE (2005)
28. Shishkov, B., Dietz, J.L.: Deriving use cases from business processes, the advantages of demo. In: ICEIS, no. 3, pp. 138–146 (2003). https://doi.org/10.1007/1-4020-2673-0_29
29. Turkman, S., Taweel, A.: Business process model driven automatic software requirements generation. In: Shishkov, B. (ed.) BMSD 2019. LNBIP, vol. 356, pp. 270–278. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24854-3_20