POLITECNICO DI TORINO
Repository ISTITUZIONALE

Extending CMMN with entity life cycles

Original
Extending CMMN with entity life cycles / Bruno, G.. - In: PROCEDIA COMPUTER SCIENCE. - ISSN 1877-0509. - ELETTRONICO. - 121:(2017), pp. 98-105. [10.1016/j.procs.2017.11.014]

Availability:
This version is available at: 11583/2699691 since: 2018-02-13T12:50:40Z

Publisher:
Elsevier

Published
DOI:10.1016/j.procs.2017.11.014

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Elsevier postprint/Author's Accepted Manuscript

© 2017. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/.The final authenticated version is available online at:
http://dx.doi.org/10.1016/j.procs.2017.11.014

(Article begins on next page)

12 September 2026

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 121 (2017) 98–105

Procedia Computer Science

www.elsevier.com/locate/procedia

CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS / ProjMAN / HCist 2017, 8-10 November 2017, Barcelona, Spain

Extending CMMN with entity life cycles

Giorgio Bruno*

Politecnico di Torino, Italy

Abstract

In the domain of business process modeling, a case denotes a situation that requires a customized treatment, and this may take place if the case workers are entitled to decide the tasks to perform as well as their ordering. In the recent CMMN (Case Management Model and Notation) standard, a case involves both an information structure and a process. The former is patterned on the file system structure and the latter is made up of stages, which are groupings of interrelated tasks. The standard leaves some open issues, such as the determination of the performers of the tasks, and the definition of the inputs and outputs of the tasks in terms of the information items affected. To address these issues, this paper proposes an extension to CMMN in which stages represent states of information items whose types along with their attributes and relationships are defined in an information model. The benefits of the extension are illustrated by means of an example that concerns the handling of papers submitted to conferences.

© 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies.

***K**eywords:* case management; CMMN; information model; case process model; stage; life cycle.

* Corresponding author. Tel.: +39 011 0907003
E-mail address: giorgio.bruno@polito.it

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies.
10.1016/j.procs.2017.11.014

1. Introduction

The notion of case has inspired the recent CMMN (Case Management Model and Notation) standard[5]. A case is a situation that requires a customized treatment and the participants in the case are in charge of making such customization, which often relies on the possibility that the number and the order of execution of tasks can be decided by their performers. This is a sharp difference from routines, in which the choice of the paths in the process is carried out automatically on the basis of predefined rules. The standard notation for routines is BPMN[1].

A case involves both a process and an information structure on which the tasks defined in the process operate. CMMN assumes that the information (documents and data) needed by a case is collected in a hierarchical structure called case file; the access to the components of the case file (called case file items) is granted to the participants in the case on the basis of the roles they play. In CMMN the case process is made up of stages which are groupings of tasks; the opening and closing of stages are based on events, such as the completion of a task, the achievement of a milestone, a change in the case file, a time event or a human decision. The notion of discretionary task is stressed as a means to introduce flexibility in the process.

However, the standard leaves some open issues, such as the determination of the performers of the tasks, and the definition of the inputs and outputs of the tasks in terms of the case file items affected. To address these issues, this paper proposes an extension to CMMN in which stages represent states of information items: a stage is then part of the life cycle of the information items of a given type. In fact, recent research in the domain of the artifact-oriented perspective[9] has stressed that the information structure of a process may include components having their own life cycles. Representing these life cycles improves the understanding of the overall model in that the life cycle of the case builds upon the life cycles of its components[4].

The proposed extension relies on the introduction of an information model that shows the types of the information entities making up the structure of the case; attributes of types and relationships between types are also specified. In addition, the information model includes the roles involved in the process and the relationships between role types and information types.

The benefits of the extension are as follows. The association of a stage with an information type means that at run time the stage will be related to a number of entities of that type; this group of entities, referred to as scope of the stage, provides the inputs for the tasks included in the stage. In addition, the input entities can be put in relation with the performers of tasks by leveraging the relationships between information types and role types.

This paper is organized as follows. Section 2 is about the related work. Section 3 presents an example that concerns the handling of papers submitted to conferences. Section 4 illustrates the top stage of the process while section 5 addresses the inner stages. Section 6 contains the conclusion.

2. Related work

The notion of case management has evolved over years[11] and several flavors have been proposed as documented by a number of recent surveys[6,8,13,14].

The main goal is to address knowledge-intensive processes where flexibility is of paramount significance. Flexibility means that the execution of tasks can be decided by the case workers, and is not subjected to a rigid control flow as it occurs in routines, which are best addressed by BPMN. Flexibility also means the capability to react to changes affecting the underlying information structure of the case. Various approaches have been compared[12] on the basis of eight key performance indicators[7].

The recent standard CMMN[5] has its roots in the Guard-Stage-Milestone approach[10] (GSM) which is data-centric and as such it emphasizes the business entities involved in the process and their life cycles[9]. The major components of GSM are the artifacts, which contain informational aspects (attributes and associations), life cycles and coordination items (events and rules). While the approach is very flexible, its major drawback is the difficulty to figure out what the actual flow of activities is. Moreover, tasks are not first-class citizens in that they are intermediated by calls to external services.

CMMN provides a simpler notation than GSM but data are not included in the process model. Therefore, the data flow is not included in the process model and the main function of stages is to decompose the model into groups of interrelated tasks.

The purpose of the extension illustrated in this paper is to reintroduce the life cycles of the major components of the case structure so as to show the data flow affecting the tasks. The explicit representation of the dataflow in business process models has also been discussed in previous papers[2,3] of the author.

3. Description of the example

This section presents the case that will be used to illustrate the proposed extension. It concerns the handling of papers submitted to a conference and is divided into a number of phases (stages) as follows. In the submission period, authors may submit papers (up to 2); they may also update or withdraw their papers. Each paper is then assigned by the chair to three reviewers and when all the assignments have been fulfilled, the chair decides the acceptance or rejection of papers. The authors of the accepted papers are in charge of providing the final versions of the papers and of registering them. Only the papers registered will be included in the conference.

At the beginning, there is a setup phase in which the chair may enter the basic information on the conference and the important dates. As the case goes on, the chair can modify the dates, update the information on the conference as well as appoint the reviewers. The important dates establish the beginning (d1) and the end (d2) of the submission period, the deadlines for the provision of reviews (d3), the assessment (d4) and the registration (d5) of papers.

The structure of a case is defined by means of an information model, which is basically a UML class model[16]. The information model of the example is shown in Fig. 1.

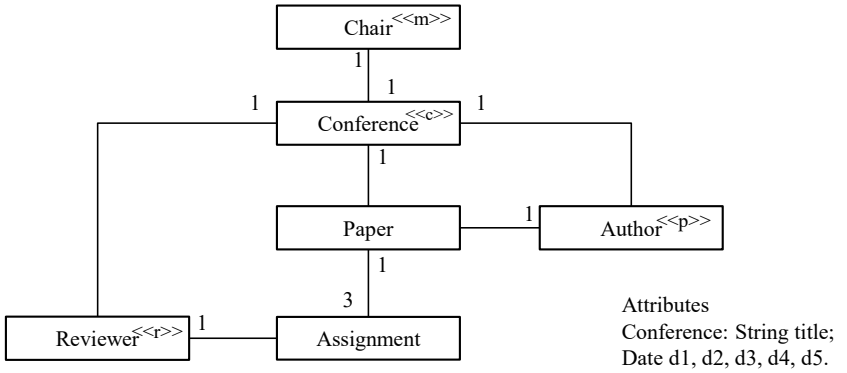

Figure 1. Information model of the example

Classes represent information entities or participant roles; stereotypes are used to assign specific meanings to them. The primary class (marked by stereotype <<c>>) represents the primary entity of the information structure: all the others classes are connected either directly or indirectly to the primary class. As a consequence, at run time all the entities different from the primary entity will be connected either directly or indirectly to the primary entity. The primary class is Conference and contains several attributes: for simplicity, only the five important dates and the conference title are considered. The other information classes are Paper and Assignment.

The role classes are Chair, Reviewer and Author. Three kinds of roles are considered: the case manager role (whose stereotype is <<m>>), the public roles (<<p>>) and the internal roles (<<r>>). All the participants in a case are represented by role entities that are associated with the primary entity. During the evolution of the case they may get associated with other information entities as well; for instance, authors are linked to the papers they entered, and reviewers to their assignments. The chair, who plays the case manager role, is associated with the primary entity when the case is instantiated. The reviewers, who play an internal role, are involved in the case by the chair. The

authors, who play a public role, must first join the case (this is a system operation) and then they can perform the tasks related to the Author role.

4. The top stage of the process

The case process is made up of stages hierarchically organized. The top stage represents the complete process. A stage has three major attributes: its name, the name of the information class whose life cycle it is part of and an optional timing constraint. Stages are depicted as rectangles with rounded corners. The top stage of the example is shown in Fig. 2. It includes six sub-stages, 5 collapsed and one expanded. If the stage is collapsed, its name appears inside the icon. If an information class name is missing in a sub-stage, it is assumed to be equal to the one of the enclosing stage; therefore, all the sub-stages in Fig. 2 are associated with the Conference class.

Timing constraints are used to specify the start date and the end date of stages. For example, the stage *paper submissions* takes place in the period established by the important dates d1 and d2. If the end date is missing, the stage is closed as a consequence of an ending condition indicated in the stage. If the start date is missing, the stage is started when the preceding stage has been closed.

The scope of a stage is the entity (or the group of entities) forming the inputs of the tasks included in the stage. The scope of the top stage is the primary entity, i.e., a conference entity in the example under consideration. When a new instance of process *handling paper submissions* is started by the case management system, it is associated with the primary entity, i.e., a conference entity. The conference entity may have been initialized before with a setup operation; for example, the chair can enter the important dates as well as the basic information on the conference through the setup operation.

Stage *updates* is shown in the expanded form: it contains three tasks to be carried out by the chair. Tasks are depicted as rectangles with their names inside. The name is followed by the execution mode: it establishes whether the task is mandatory or optional and whether it may be repeated or not. The execution mode takes one of these three forms: a range (l..h), character ‘n’ or character ‘*’. The lower limit of the range may be 1 (if the task is mandatory) or 0 (if it is optional); the higher limit is an integer number >= l. Character ‘n’ (‘*’) means that the task is mandatory (optional) and may be repeated a number of times that is not predetermined. If the execution mode is missing, the task is mandatory and non-repeatable.

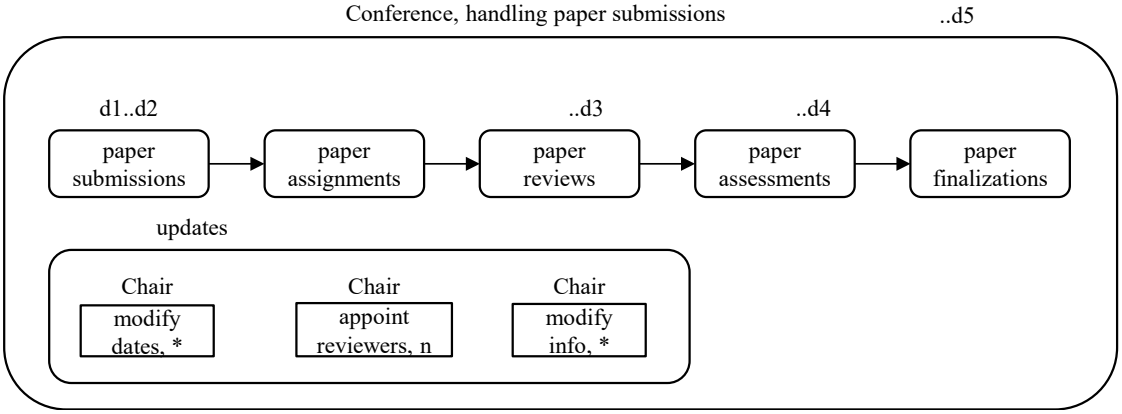

Figure 2. The top stage of the example

The role, e.g. Chair, of the performers entitled to perform the tasks is shown by the label near the task icons.

The sub-stages of the top stage form two concurrent paths. A path is a connected graph made up of stages: it has exactly one initial stage, i.e., a stage with no input arcs, and one or more completion stages (i.e., stages having no

output arcs). The output arcs point to the next stages; if a stage is followed by two or more stages, conditions are associated with the output arcs so that a choice can be made when the stage finishes.

A stage finishes when the end date has expired or the ending condition has become true; the ending condition, if any, is associated with the stage. When the top stage is instantiated, it is associated with the primary entity of the case and becomes active. As a consequence, the initial stages of its inner paths become potentially active: they are immediately active if they have no starting dates or they will become active at their starting dates. Sub-stage *updates* is immediately active, while *paper submissions* will be active on the day established by d1 (an attribute of the primary entity). When a stage is active, all the tasks and stages included become potentially active.

The upper path, referred to as the Paper path, is the main one and shows how submissions are handled. The other path consists of task *updates* which enables the chair to modify the dates or the information on the conference while the main path is going on. In addition, the chair can add reviewers to the case.

Tasks can operate on the entities belonging to the scope of the enclosing stage. For example, the post-condition written in natural language of task *appoint reviewers* could be as follows: a number of Reviewer entities have been generated and connected to the conference entity. The term "conference entity" denotes the entity in the scope of stage *updates*; it is the input entity of the task.

Tasks are performed by the participants in the case who play the role required and are associated with the input entity (or entities). The chair who can perform the tasks in stage *updates* is the one associated with the conference.

5. Inner stages

The expansion of the first three stages of the Paper path is shown in Fig. 3.

When *paper submissions* becomes active, task *enter paper* and sub-stage *entered* get enabled. Stage *entered* is the initial stage of the Paper life cycle; its scope is initially empty.

Task *enter paper* is a generative task in that its effect is to add a new paper entity to the case; this is shown by the label *new* attached to its output arc. Whenever *enter paper* is performed, a new paper becomes part of the scope of stage *entered*. This stage contains two tasks, *modify paper* and *withdraw paper*, which enable the authors to act on their papers. Task *modify paper* does not change the stage of the paper, while *withdraw paper* brings the paper into stage *withdrawn*, which is a final stage of the Paper life cycle. A final stage is shown collapsed in that it contains no tasks and is marked with a small black circle.

If a paper is not withdrawn, it remains in stage *entered*, which is not a final stage. However, when the enclosing stage *paper submissions* finishes, all the papers in stage *entered* need to be moved into the next stage of the Paper life cycle. The next stage is named *submitted* and is not managed in the current container: for this reason, it is shown collapsed in that its tasks are not active in the current container and is marked with a small white circle. Stage submitted is managed in stage *paper assignments*. An arc from a normal stage (i.e., a stage that is neither a final stage nor an unmanaged one) to an unmanaged stage, such as the arc from *entered* to *submitted,* shows the connection between two stages (of the same life cycle) belonging to different containers.

As to the performers of the tasks in stage *paper submissions*, task *enter paper* may be carried out by the authors associated with the conference entity (the input of the task). A new author must first join the conference (and this is a feature of a public role such as Author) and then they will be entitled to enter a paper; as a consequence the newly generated paper will be associated with its author. The post-condition of the task is as follows: a new paper has been generated and connected to the conference and the author.

While the scope of stage *paper submissions* includes only the conference entity, the scope of stage *entered* consists of the paper entities produced by task *enter paper*. The papers are associated with their authors and therefore each execution of tasks *modify paper* and *withdraw paper* acts on a paper entity and is carried out by the author associated with that entity; this rule leverages the relationships that are established in the information model between information types and role types.

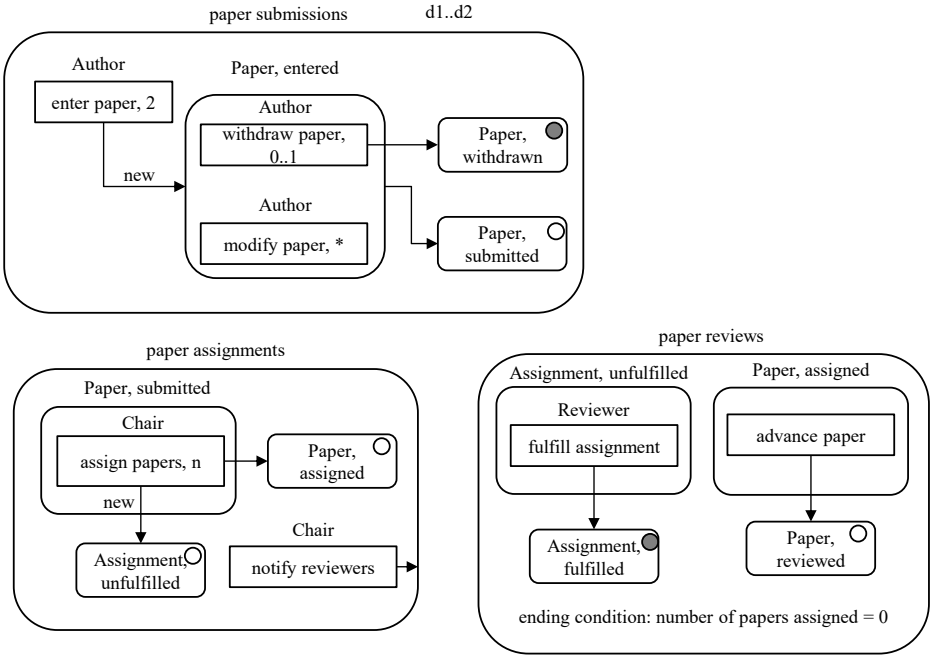

Figure 3. The first three stages of the Paper path

When deadline d2 expires, stage *paper submissions* ends and stage *paper assignments* becomes active. Sub-stage *submitted* becomes active and task *assign papers* can act on the papers included in the scope of the stage: such papers were collected when *submitted* was an unmanaged sub-stage of stage *paper submissions*.

The chair can assign papers to reviewers by producing assignments. Although the Chair entity is not directly linked to the paper entities, it is connected to them indirectly through the conference entity. To clarify the indirect connection, the derived relationship Paper - Chair = Paper - Conference - Chair can be added to the information model.

In complex situations, a performer rule can be added to determine the intended performer of a task; in natural language, the rule for *assign papers* could be as follows: the performer of the execution of the task on the input papers is the chair of the conference the input papers belong to.

Task *assign papers* generates three assignments for each input paper. The papers are then moved into the unmanaged stage *assigned* and the newly generated assignments enter stage *unfulfilled*, which is the initial stage of the Assignment life cycle.

Task *notify reviewers* enables the chair to inform the reviewers, through emails, that the assignments are ready. It is subjected to the precondition that there are no more papers in stage *submitted*. The execution of this task ends the stage: this consequence is expressed by an output arc with no destination. Such an arc is a graphical representation of the ending condition related to the completion of a task.

Stage *paper reviews* has two concurrent paths concerning the Assignment life cycle and the Paper one. Task *fulfill assignment* enables reviewers to fulfill their assignments and moves the assignments into the final stage *fulfilled*. Task *advance paper* is an automatic one in that it has no role associated. It moves a paper having all its three associated assignments fulfilled from stage *assigned* to stage *reviewed* which is an unmanaged stage.

The stage finishes when the ending condition becomes true; this occurs when the number of papers which are still in stage *assigned* amounts to zero.

The last two stages of the Paper path are shown in Fig. 4.

Stage *paper assessments* enables the chair to either accept or reject papers and then to inform the authors with emails. Task *notify authors* is an ending task like task *notify reviewers* in stage *paper assignments*; its precondition is that the number of papers still in stage *reviewed* is 0.

The last stage, *paper finalizations*, allows authors to provide the final versions of accepted papers and then to register the finalized papers in the conference. Task *enter registration* moves a finalized paper into stage *registered*. The stage ends when the container stage ends, i.e., when deadline d5 expires. If there are still papers in stages *accepted* or *finalized*, they are automatically moved into the final stage *incomplete*. Only the registered papers will be included in the conference.

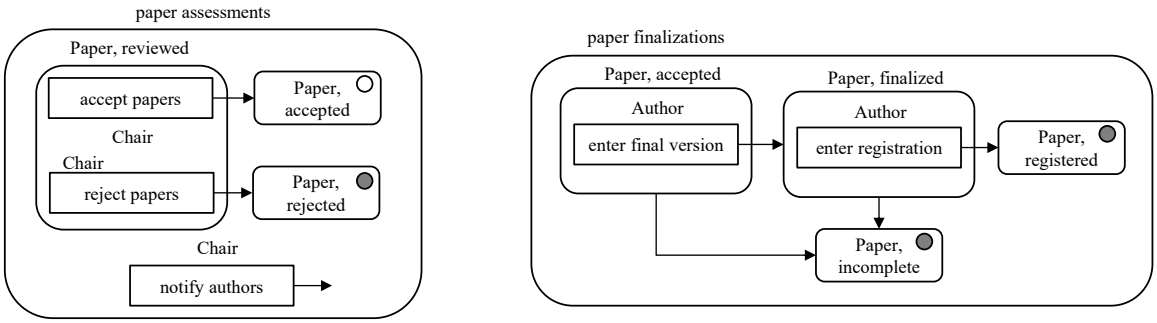

Figure 4. The last two stages of the Paper path

6. Conclusion

This paper has elaborated on the notion of stage, which CMMN mainly uses to provide a functional top-down decomposition, thus losing the lifecycle orientation of the GSM approach it has been inspired by.

An extension has been proposed in which stages are associated with entity types, such as Conference or Paper. The stages referring to the same entity type define the life cycle of such entities. Tasks are shown inside stages and then they take the entities in the stages they belong to as the input ones. The stages of the output entities are indicated by the output links of the tasks.

The entity types along with their relationships and attributes are shown in a companion information model: this enables the pre and post-conditions of the tasks to be easily expressed.

The overall structure of the process retains the hierarchical approach of CMMN with the difference that the stages indicate which information entities they are related to. The process results from the interweaving of the life cycles of the entity types of the case.

Further work is developing in two directions. The aim of one direction is to define the work structure of the participants in terms of a workspace instead of a work list. The purpose is to give participants more visibility on the case and this was the objective of case handling[17], a precursor of case management.

The other direction is to evolve CMMN towards social business process management[15]. One of the issues is to improve the collaboration among participants so that they can attain a common understanding of the problem under consideration through the tools provided by the case management system.

References

1. BPMN, Business Process Model and Notation, V.2.0.2. Retrieved February 4, 2017, from http://www.omg.org/spec/BPMN/2.0.2/
2. Bruno, G.: Data flow and human tasks in business process models. Procedia Computer Science, vol. 64, pp. 379–386 (2015)
3. Bruno, G.: Tasks and assignments in case management models. Procedia Computer Science, vol. 100, pp. 156–163 (2016)
4. Chao, T., et al.: Artifact-based transformation of IBM Global Financing. LNCS, vol. 5701, pp. 261–277. Springer, Heidelberg (2009)
5. CMMN. Case Management Model and Notation, V.1.0. Retrieved February 4, 2017, from http://www.omg.org/spec/CMMN
6. De Man, H.: Case management: a review of modeling approaches. BPTrends, January (2009).
7. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. Journal on Data Semantics, 4, 29–57 (2015)
8. Hauder, M., Pigat, S., Matthes, F.: Research challenges in adaptive case management: a literature review. In IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), pp. 98–107 (2014)
9. Hull, R.: Artifact-centric business process models: brief survey of research results and challenges. LNCS, vol. 5332, pp. 1152–1163. Heidelberg: Springer (2008)
10. Hull, R., et al.: Introducing the Guard-Stage-Milestone approach for specifying business entity lifecycles. LNCS, vol. 6551, pp. 1–24. Springer, Heidelberg (2011)
11. Marin, M., Hauder, M.: Case Management: a data set of definitions. Cornell University Library: arXiv:1507.04004v1 (2015)
12. Marin, M., Hauder, M., Matthes, F.: Case management: an evaluation of existing approaches for knowledge-intensive processes. In 4th International Workshop on Adaptive Case Management and other non-workflow approaches to BPM. Springer, Heidelberg (2015)
13. Marin, M., Hull, R., Vaculín, R.: Data centric BPM and the emerging case management standard: a short survey. LNBIP, vol. 132, pp 24–30. Springer, Heidelberg (2013)
14. Motahari-Nezhad, H. R., Swenson, K. D.: Adaptive case management: overview and research challenges. In IEEE 15th Conference on Business Informatics, pp. 264–269 (2013).
15. Pflanzl, N., Vossen, G.: Challenges of social business process management. In IEEE 47th Hawaii International Conference on System Sciences, pp. 3868–3877 (2014).
16. UML. Unified Modeling Language, V.2.4.1. Retrieved February 4, 2017, from http://www.omg.org/spec/UML/2.4.1/
17. Van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data & Knowledge Engineering, vol. 53 (2), pp 129–162 (2005)