Adaptive Case Management as a Process of Construction of and Movement in a State Space

Ilia Bider[1,2], Amin Jalali[1], Jens Ohlsson[1]

[1] DSV, Stockholm University, Stockholm, Forum 100, SE-16440 Kista, Sweden
[2] 2IbisSoft AB, Stockholm, Box 19567, SE-10432 Stockholm, Sweden
ilia@{dsv.su|ibissoft}.se; {aj,jeoh}@dsv.su.se

Abstract. Despite having a number of years of experience, adaptive case management (ACM) still does not have a theory that would differentiate it from other paradigms of business process management and support. The known attempts to formalize Case Management do not seem to help much in creating an approach that could be useful in practice. This paper suggests an approach to building such a theory based on generalization of what is used in practice on one hand and the state-oriented view on business processes on the other. In practice, ACM systems use a number of ready-made templates that are picked up and filled as necessary for the case. State-oriented view considers a process instance/case as a point moving in a specially constructed state space. This paper suggests considering a case template as a definition of a sub-space and piking different template on the fly as constructing the state space along with moving in it when filling the template. The result is similar to what in control-flow based theories are considered as a state space with variable numbers of dimensions. Beside suggestions to building a theory, the paper demonstrates the usage of the theory on an example.

Keywords: Adaptive Case Management, State Space, Business Process

1 Motivation

Adaptive case management (ACM) as a practical discipline has emerged for some years ago [9]. However, it still lacks a theory or a model that could explain what an ACM system is that could be used for analysis, comparison and development of such systems. Moreover, in our view, there is no commonly accepted theory of case management (CM) or case handling systems, even non-adaptive ones. The goal of this idea paper is to suggest an approach to building a theory/model of ACM systems.

As both CM and ACM systems are aimed at supporting workers (so called knowledge workers in case of ACM systems) in driving their working/business processes, naturally, both CM and ACM systems falls in the category of Business Process Support (BPS) systems, and thus belong to the wider domain of Business Process Management (BPM). In contemporary BPM, the predominant view on business processes is the activity or task oriented one. More exactly, a business process instance or case is considered as a set of activities aimed at reaching some goal. Business process type or model is considered as a set of rules that

determine which activities to be executed when running a process instance that belong to the particular type. This view is considered the as a cornerstone when developing BPS systems.

Traditionally, BPS systems are built based on a workflow model of business processes they aim to support. A workflow model is a kind of a graph that connects activities between themselves and thus determines what activities should/could be executed after finishing a particular one. From the point of view of the activity-oriented paradigm, the natural way of developing a theory/model of CM systems is to continue using activities as a primary concept of the theory/model, but drop the idea of being able to represent the connections between them with the help of a graph. The initiative of picking up the next activity to execute is left in the hands of humans, while the system assists the execution of an activity after it has been picked up. An attempt of formalizing this approach has been suggested in [1]. To the best of our knowledge, there were no comprehensive attempts to build a theory for CM and/or ACM systems since van der Aalst et al, [1] work has been published.

The assumption of the concept of activity/task being primary and even mandatory in a theory/model of ACM systems contradicts our experience of business analysis and design of case handling/management systems. Our experience started with analysis of case handling in Swedish municipalities that included general case handling in the Municipality of Motala [3], and case handling in the social welfare office of municipality of Jönköping, e.g. handling applications for child adoption, and later handling cases of suspected child abuse. Case handling that we observed in municipalities were rather template/form driven, than activity driven. There were templates, often of the type of structured forms, for an application, investigation, and decision making.

Templates/forms driven case handling cannot easily be translated into the activities driven one, as the same template/form can be used for several different activities, e.g., for both application and decision making. Activities can be represented in such case handling in different ways. For example, an activity of decision making can be represented in the form by three fields: (a) decision, (b) name of decision maker and (c) date when the decision has been taken. Another example, a set of activities can be represented as a checklist on the form that requires putting a cross beside each activity on the list before the form can be considered as properly filled.

In addition, there are often no strict rules on when starting to work with a certain template. For example, the main result of an adoption case is a report that follows a certain template. The report is based on a series of meetings with the applicant(s), but there is no regulation when to start writing the report, directly after the first meeting or, after the whole series has been completed. The choice is left to individual case managers. The standardization of case handling in municipalities is also done not by specifying activities, but by producing a package of templates/forms mandatory to be used during handling cases. This type of standardization was, for example, used by Swedish National Board of Health and Welfare for handling cases of suspected child abuse. This process was used as a pilot process when we created our tool for building case-oriented BPS systems iPB [4].

Summarizing our experience, a theory/model of CM systems, including ACM systems does not need to be based on the notion of activity/task as a primary concept. In a simplified form, it does not need to have it at all. It would be enough if the theory can explain the concept of templates/forms, and filling them during the lifespan of each case. The latter can be done using the state-oriented view on business processes [8]. In the state oriented view, a process instance is considered to be a trajectory in a specially constructed state-space, while a process type/model is considered as a set of restrictions on the trajectories of instances that belong to the given type. From this point of view, templates/forms are used to represent the structure of the chosen state space to the end users of a CM system, while filling them represents movement of the instance in the state space. Restrictions on the movement can be defined, for example, by demanding finishing filling one form before starting with another one [4].

The state-oriented view suggested in [4, 8] serves well as a basis for the theory of CM, but it does not naturally fit the reality of ACM. The works cited above assume that the state-space is the same for all cases which corresponds well to CM with few pre-defined templates/forms. An ACM system may include many different templates/forms from which the workers pick some for a particular case dependent on the needs. This makes it artificial to consider each instance/case of a given process type having the same state-space. The state-oriented view needs to be extended to consider a possibility of state-space being constructed while a case is in progress. The goal of this paper is to outline the idea of using an evolving state space for creating a theory/model that can be used for analysis, comparison and development of ACM systems.

The discussion of the idea is done through analysis of an example a process for preparing and giving a course in a university borrowed from [6]. The rest of the paper is structured in the following way. In Section 2, we give a short overview of the state oriented view on business processes from [8]. In Section 3, we describe an example of an ACM type of business processes. In Section 4, based on this example, we discuss how the example can be interpreted from the viewpoint of the evolving state space. Section 5 contains concluding remarks and plans for the future work.

2 State oriented view of business processes

We suggest using the state-oriented view on business processes as a foundation for building a theory of ACM. As this is not a standard view, in this section, we give a brief overview of its underlying concepts and principles as suggested in [8]. The origin of the state-oriented view on business processes lies outside the business process domain. The idea comes from the Mathematical System Theory and especially the theory of hybrid dynamical systems [10]. Another source used for developing this theory was an objects-connectors model of human-computer interaction developed in [5]. In essence, the state-oriented view on business processes is an application of the ideas worked out for modeling and controlling physical processes to the domain of business processes. The main concept of the state-oriented view is a state of the process instance that can be defined as a position in some state space. A state space is considered multidimensional, where

each dimension represents some important parameter (and its possible values) of the business process. Each point in the state space represents a possible result of the execution of a process instance. If we add the time axis to the state space, then a trajectory (curve) in the space-time will represent a possible execution of a process instance in time. A process type is defined as a subset of allowed trajectories in space-time. As an example, consider an order process from Fig. 1. Its state space can be presented as a set of numeric dimensions from Fig. 2 defined in the following way:

- In the first place there are a number of pairs of product-related dimensions <ordered, delivered>, one pair for each product being sold. The first dimension represents the number of ordered items of a particular product. The second one represents the number of already delivered items of this product. The number of such pairs of dimensions can be considered as equal to the size of the companys product assortment.
- In addition, there are two numeric dimensions concerning payment: invoiced (amount of money invoiced) and paid (amount of money already received from the customer).
- Each process instance of the given type has a goal that can be defined as a set of conditions that have to be fulfilled before a process instance can be considered as finished (i.e. end of the process instance trajectory in the space state). A state that satisfies these conditions is called a final state of the process. The set of final states for the process in Fig. 2 can be defined as follows: (a) for each ordered item Ordered = Delivered; (b) To pay = Total + Freight + Tax; (c) Invoiced = To pay; (d) Paid = Invoiced. These conditions define a surface in the state space of this process type.

The process instance is driven forward through activities executed either automatically or with human assistance. Activities can be planned first and executed later. A planned activity records such information as type of action (goods shipment, compiling a program, sending a letter), planned date and time, deadline, name of a person responsible for an action, etc. All activities planned and executed in the frame of the process should be aimed at diminishing the distance between the current position in the state space and the nearest final

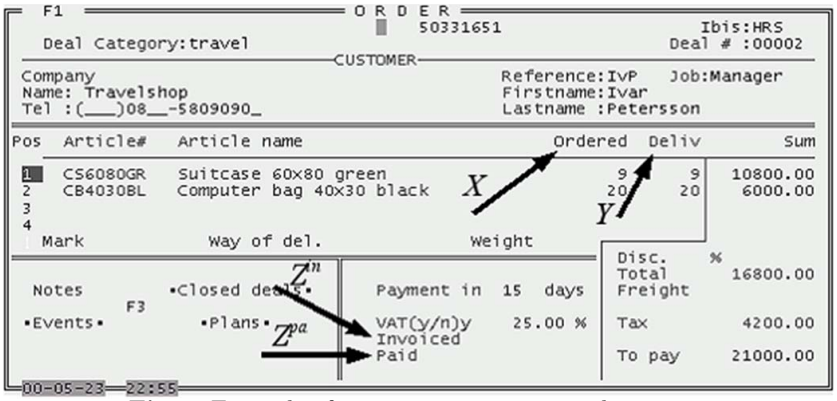

Fig. 1. Example of a process state as a mockup screen

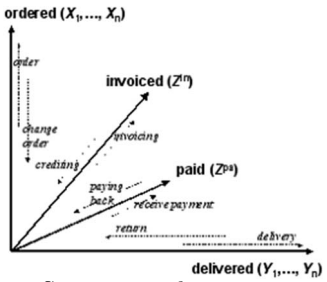

Fig. 2. State space dimensions as axes

state. The meaning of the term distance depends on the business process in question. Here, we use the term informally. For example, activities to plan for the process in Fig. 1 can be defined in the following manner:

- If for some item Ordered > Delivered, shipment should be performed, or
- If To pay > Invoiced, an invoice should be sent, etc.

All activities currently planned for a process instance make up the process plan. The plan together with the current position in the state space constitutes a so-called generalized state of the process, the plan being an active part of it (engine). The plan plays the same role as the derivatives in the formalisms used for modeling and controlling physical processes in Mathematical systems theory. The plan shows the direction (type of action) and speed of movement (deadlines), just as first derivatives do in a continuous state space. Using the concept of process plan, the control over the process instance can be defined in the following manner. First, an activity from the plan is executed and the position of the process instance in the state space is changed. Then, depending on the new position, the plan is corrected, i.e., new actions are added to the plan and some existing ones are modified or removed from it. Correction of the plan can be done manually by the process participants, automatically by a system that supports the process, or in a mixed fashion [3]. The iterations continue until the instance reaches its goal.

3 An example a course process at a university

An example that we will explore when discussing our idea is concerned preparing and giving a course occasion at a university, as presented in [6] [4]. Such an occasion can be the first one of a newly developed course, or the next occasion of an already existing course. So far, only a prototype of a system that supports this process has been built. Nevertheless, we decided to consider this process in

[3] The mixed fashion of planning can be used for regulating the level of flexibility/rigidity allowed for a process type. Automatic rules gives rigidity, freedom for manual planning gives flexibility. A method of regulating the level of flexibility of planning based on deontic logic is discussed in [7]

[4] Bider et al. [6] is an experience report that describes the project of building a model of a course preparation process. It does not discuss any theoretical aspects presented in this paper.

our paper and not the one for which a real system is already in operation due to the following reasons:

- The domain is familiar to both academics, who gives courses (though they may not agree with how it is done in our department), and to practitioners who were students at some time in the past. This gives an opportunity to describe the process in a concise manner in the paper of a limited size. Using a process for which a real system is in operation would require becoming familiar with the business of social offices in Swedish municipalities.
- The process has all elements we need for presenting our approach to building a theory of CM/ACM systems.

The templates/forms to be used in instances/cases of the course preparation process are presented in Fig. 3, an example of a template, Lecture/Lesson, is shown in Fig. 4. As follows from Fig. 3, the templates are grouped in two categories templates for preparing a course and templates for gathering feedback from the course. The first group consists of templates: Course general description, Course book, Course compendium, Article compendium, Lecture/Lesson, Seminar, Lab and Exam. The second group consists of the rest of the templates. While preparing a course occasion, the teacher(s) decides on which teaching/learning activities, and which material will be used in the course occasion, picks up a form for each activity or material and fills it in. Naturally, some forms can be employed only once, e.g., Course general description, or Exam, others can be employed several times, or none at all. Though the templates are presented in Fig. 3 in a certain order, this order does not enforce the usage of templates. For example, on one occasion all preparation can be finished before the actual course starts. In another occasion, only the first introductory lecture is prepared before the start, all other teaching/learning activities are prepared and completed while the course is running based on the feedback both from the teachers and the students. The teachers can freely choose the templates for preparation and change information in them at any moment before the actual teaching/learning activity takes place, including deleting some of them. However this is not true for the templates aimed at gathering feedback. These templates should be synchronized with the ones already chosen for teaching/learning activities. For example, if two lectures have been chosen for the course, four feedback forms should be automatically selected for the course occasion: two of the type Lecture/Lesson teacher

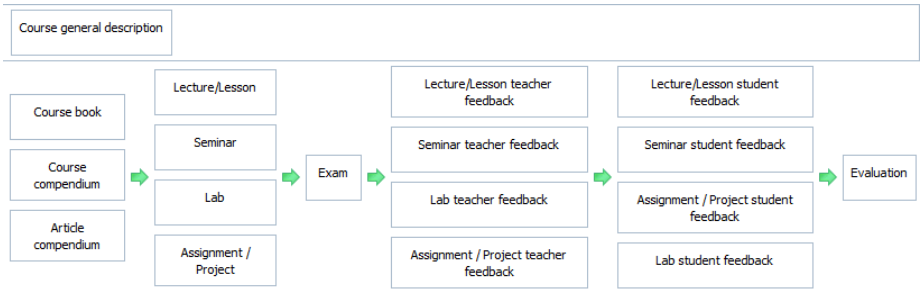

Fig. 3. Templates for course process

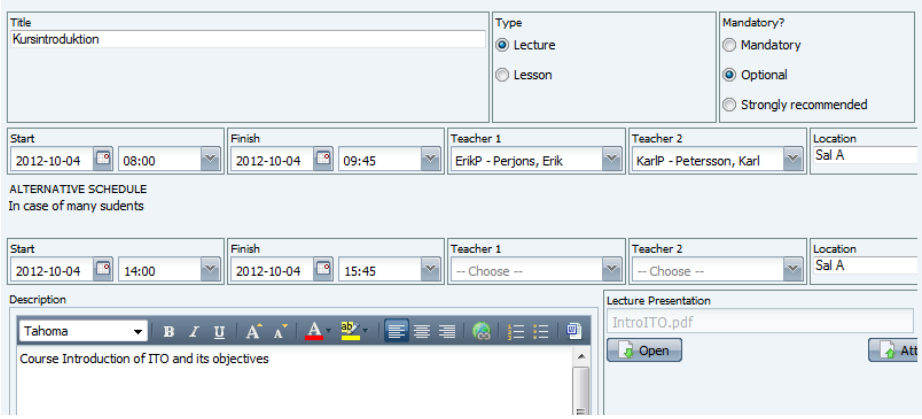

Fig. 4. Template/form for Lecture/Lesson box in Fig. 3

feedback and two of the type Lecture/Lesson student feedback, see Fig. 3. An example for a Lecture/Lesson teacher feedback form is presented in Fig. 5; the figure represents a form synchronized with the one shown in Fig. 4. As we see from Fig. 5, part of the fields in this form (the upper part of the form) are inherited from the form with which the current form is synchronized.

Summarizing the example presented, a template/form driven ACM system to support a course process needs to provide the following basic functionality:

- Repository of templates/forms that can be chosen for attaching to an instance/case
- Navigation through the forms already attached to the instance/case
- Manipulating the templates/forms already attached to the instance/case fill with new data, update the content, delete, marked as finished/closed
- Automatically adding synchronized forms when they are needed
- Providing restrictions on possibility to attach a given template/form to a case based on what other templates/forms have already been attached and their status (e.g. finished).

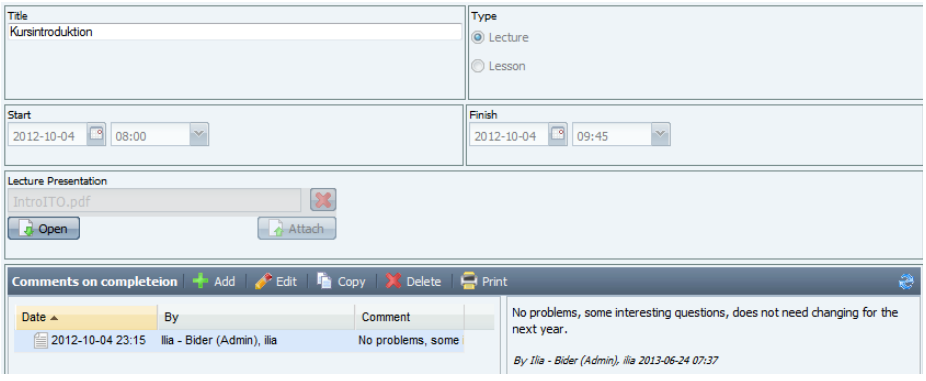

Fig. 5. Template/form for Lecture/Lesson feedback box in Fig. 3 synchronized with the form in Fig. 4

An example of how the above functionality can be presented to the end user is shown in Fig. 6 [4]. Fig. 6 represents an upper level view of an instance/case. Blue and green rectangles indicate templates/forms already attached to the case. Blue color is used to show that all templates/forms behind the rectangle are already filled with information considered to be required and/or sufficient. The numbers below the name of the template shows how many templates/forms of this type have already been attached to the case (number 1 is not showen) White color represents the forms that the end user is free to attach to the case, grey color represents the forms that cannot be attached to the case due to some restrictions.

A diagram in Fig. 6 is used for navigation between the forms. By clicking on a rectangle the user goes to a place where he can manipulate the templates/forms of the type the rectangle represents, e.g., create a new form, add information, change information already entered, delete a form or read what has been entered by others.

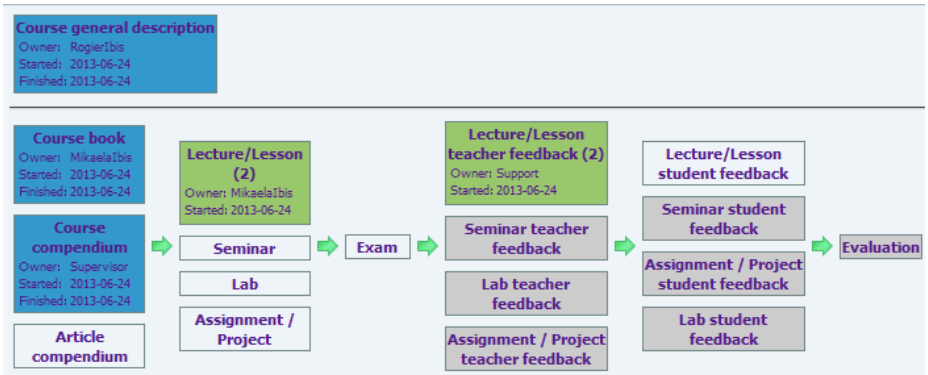

Fig. 6. A course case in iPB

4 Interpreting the example from the state-oriented perspective

From the state-oriented view, the idea of case handling discussed in Section 3 can be interpreted in the following way:

– Attaching a new template/form to or deleting the existing template from a case can be considered as an operation of constructing a state-space for a given case. A template here represents a standard subspace that can be added to or subtracted from the case state space. For example, the template in Fig. 4 defines a subspace with the three groups of dimensions expressing different perspectives of case handling, data perspective, time perspective and resource perspective as represented in Fig. 7 [5].

[5] Actually there are more perspectives that the three ones represented in Fig. 7, for example, space dimension (Location).

– Filling a template/form corresponds to movement in the subspace defined by this template. For example assigning teachers to a lecture represents movement along r-dimensions in Fig. 7. Creating the lecture content corresponds to movement along d-dimensions in Fig. 7. Deciding on time represents movement along the t-dimensions in Fig. 7. The order in which the movement occurs can be different. In one case, a teacher is assigned first and he/she then creates the content, in another case, the content is borrowed from the previous course occasion, and then a new teacher is assigned to mediate it to the students.
– Restrictions on adding forms to a particular case represent (1) constraints on overall structure of the constructed state space, and (2) the order in which the movement is allowed to occur. The first type of restrictions corresponds to the synchronized forms; they cannot be added without first adding the basic ones. An example of the second type of restrictions can be to not allow adding lectures before choosing the course book, which will determine the content of the lectures.

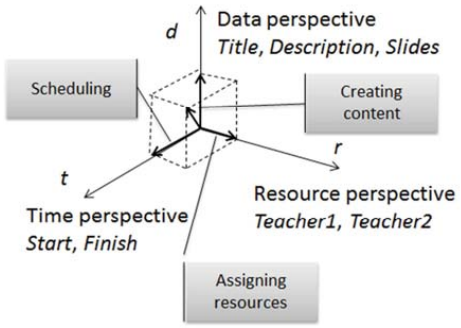

Fig. 7. A subspace that corresponds the form in Fig. 4

5 Concluding remarks

In this paper, by use of an example, we showed that it is possible to build a CM and ACM system without explicitly introducing the notion of activity/task. This exercise was not aimed at excluding this notion from the world of CM and ACM altogether, as this notion can be useful in making CM and ACM systems more helpful when supporting people in running their process instances/cases. Our main position is not to reject the notion of activity, but show that this notion is secondary to other notions. We also suggested a theoretical underpinning of the template/form driven CM and ACM systems in the form of the state-oriented view suggested in [8]. For ACM system, this view has been extended to include possibility of the state-space being constructed while a process/instance develops in time. This construction is completed by adding new sub-spaces based on the needs arising in the particular process instance/case.

Based on our suggestions it is possible to analyze the functionality of a CM or ACM system to see whether it allows all operations listed in Section 3 and interpreted in Section 4. An immediate result of applying this analysis to our own tool iPB (Bider et al. 2010) showed that while this tool is quite suitable for building CM systems, it becomes cumbersome when applied to ACM systems. One of the main reasons to it is the decision to have all possible templates/forms presented in the case navigation panel of Fig. 6, independently of whether these forms can be used in the given instance/case or not. A better decision would be to have the not used forms outside the panel until they are selected by a user.

Our future plans in regards to this work lie in two directions. One of these directions is full formalization of the ideas presented in the paper; the other one is creating a practical methodology for analysis of CM and ACM systems. One way of approaching the second task could be via defining patterns from which a practically useful state space can be created, and rules for combining them. This would be analogous to the workflow patterns suggested for the standard BPM systems. An approach for defining patterns for the state-oriented view from [2] could be used for this end.

References

1. W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.
2. B. Andersson, I. Bider, P. Johannesson, and E. Perjons. Towards a formal definition of goal-oriented business process patterns. *Business Process Management Journal*, 11(6):650–662, 2005.
3. T. Andersson, A. Andersson-Ceder, and I. Bider. State flow as a way of analyzing business processes–case studies. *Logistics Information Management*, 15(1):34–45, 2002.
4. I. Bider, P. Johannesson, and E. Perjons. In search of the holy grail: Integrating social software with bpm experience report. In *Enterprise, Business-Process and Information Systems Modeling*, pages 1–13. Springer, 2010.
5. I. Bider, M. Khomyakov, and E. Pushchinsky. Logic of change: Semantics of object systems with active relations. *Automated Software Engineering*, 7(1):9–37, 2000.
6. I. Bider, E. Perjons, and Z.R. Dar. Using data-centric business process modeling for discovering requirements for business process support systems: Experience report. In *Enterprise, Business-Process and Information Systems Modeling*, pages 63–77. Springer, 2013.
7. I. Bider and A. Striy. Controlling business process instance flexibility via rules of planning. *International Journal of Business Process Integration and Management*, 3(1):15–25, 2008.
8. M. Khomyakov and I. Bider. Achieving workflow flexibility through taming the chaos. In *OOIS 2000*, pages 85–92. Springer, 2001.
9. K Swenson. Mastering the unpredictable: How adaptive case management will revolutionize the way that knowledge workers get things done; meghan, 2010.
10. A.J. van der Schaft and J.M. Schumacher. *An introduction to hybrid dynamical systems*, volume 251. Springer London, 2000.