Towards a pattern recognition approach for transferring knowledge in ACM

Thanh Tran Thi Kim
Christoph Ruhsam
Max Pucher
ISIS Papyrus Europe AG
Austria

Maximilian Kobler
University of Applied Sciences
Burgenland
Austria

Jan Mendling
Wirtschaftsuniversität Wien,
Institute for Information Business
Austria

***Abstract*— In Adaptive Case Management (ACM) systems, knowledge workers have the flexibility to deal with unpredictable situations. Compared with a classical BPM approach the extensive prescriptive process analysis and definitions are replaced by context-sensitive proposals, which is more suited for knowledge-intensive work. Thus, it is vital that ACM systems support knowledge workers with knowledge captured from previous work which can be ambiguous for the system. This paper proposes an approach to support knowledge workers based on the knowledge previously applied by others in the form of a User Trained Agent that learns from ad hoc actions taken by knowledge workers to suggest best next actions for the current situation. The proposed best next actions are analyzed for coherence.**

Keywords— ACM; pattern recognition; adaptive system; decision support system; UTA; user trained agent

I. INTRODUCTION

Adaptive Case Management (ACM) systems provide knowledge workers with capabilities to handle business cases in a flexible way [1, 2, 3]. Besides following the processes predefined in the systems for situations typically calling for strict workflow needs like sign-off processes, knowledge workers are able to decide adequately based on their experience and knowledge on which ad hoc actions to conduct given the current situation, irrespective of whether predefined or not [4]. Currently most ACM systems allow knowledge workers to select single ad-hoc actions freely but that action will not in any way influence future execution. There is no embedded learning mechanism availability towards facilitating reuse of such ad hoc actions. ACM systems that provide a user created template library are organized into categories but do not support the knowledge worker in their decision making at certain points in the process [5, 6]. In a specific situation the decision is made only based on the knowledge worker's own experience but not from the collective knowledge base stemming from all users working with the system. Thus, the knowledge "which is only between two ears of a knowledge worker" [7] is not available for other knowledge workers in the ACM systems.

To overcome that lack of knowledge transfer we propose in this paper an approach for capturing knowledge within an ACM system and use that base for user recommendations. A User Trained Agent (UTA) [8, 9] is built on a pattern recognition principle [10]. A pattern represents a collection of elements relevant for a decision taken from the scope of a process or case to solve a particular problem in a certain context [11]. The UTA analyzes what elements of a pattern are relevant for an ad hoc action. Based on the gathered pattern knowledge, the UTA recommends an ad hoc action as, best next action (BNA) to knowledge workers when a pattern match is identified.

The role of the UTA in an ACM system is depicted in Fig. 1. In an ACM system, business processes can be predefined at design time and executed by knowledge workers at run time. Moreover, they can also add ad hoc tasks at run time as needed by assembling several ad hoc actions on the fly to work towards a well-defined goal. The UTA is applied in the ACM system to support knowledge workers in unpredictable situations to identify the BNA. The knowledge worker can decide whether to follow the recommendation or execute another ad hoc action. The main functionality of the UTA is to observe an ad hoc action created on the fly by knowledge workers; learn in which state an ad hoc action is created; and recommend an ad hoc action in a similar situation when its state is recognized from the knowledge of the UTA.

In this paper, we introduce the UTA principle and the benefits achieved when the UTA is applied in an ACM system for knowledge sharing in a group of business users. The UTA internals and the pattern matching algorithms are not covered within this paper. The remainder of this paper is structured as follows: Section 2 introduces the UTA mechanism in the context of an ACM system. Section 3 describes the integration of the UTA in the ISIS Papyrus ACM Solution using a central

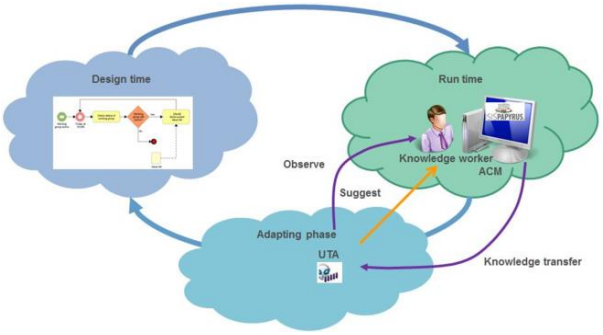

Fig. 1. The ACM environment

978-1-4799-5467-4/14 $31.00 © 2014 IEEE
DOI 10.1109/EDOCW.2014.28

metadata Repository for all case definitions. Related works are discussed in Section 4. The benefit of the approach is summarized in Section 5 as a conclusion of this paper.

II. UTA IN ACM

As mentioned above, UTA is built on pattern recognition principles [1, 9, 10]. In the context of ACM a pattern is a state of a case when an ad hoc action is created. The state of a case is defined by its system and data objects and their attributes observed by the UTA. Depending on the configuration of the UTA, the state space could be the entire data related to the case or some selected features defined by a domain specific ontology. For example, a case in ACM is composed of goals, artifacts, tasks, processes, data objects [2]. These contextual elements can be potentially observed by the UTA and used as attributes for pattern recognition in a dynamic context. There are several challenges to manage complexity in ACM and proposed approaches have to deliver proof-of-value [12, 13].

The UTA operation is composed of two main functions i.e. learning user actions related to case patterns, and recommending actions to performers when similar patterns are identified. In contrast to most machine learning programs that are trained with sample data sets, the learning function of the UTA is triggered in real-time by changes in the defined state space of the case, which are applied by the knowledge workers. The UTA extracts relevant data features to create action clusters for identifiying patterns. A learning function is triggered when an ad hoc action is taken by a particular knowledge worker. It is important that the learning is related to the role of the performer because the recommendation also has to be given to a performer with the same role. The features are extracted from the state space composed of the attributes of a case observed by the UTA. Each ad hoc action is represented by a cluster containing features and decision parameters that are used to decide whether a pattern is matched to the cluster. If a pattern matches the cluster, the action represented by the cluster is recommended to the knowledge worker and becomes part of the ongoing learning process if accepted.

Knowledge workers do not have to ask explicitly for a suggestion but the UTA will activate its recommendation function autonomously each time a pattern is identified and then recommends the action for which a pattern was found. Each time a user takes an action the UTA will trigger the learning function and look for related patterns. The recommendation function is executed in real time by scanning the knowledge of the UTA. The UTA's knowledge can stem from diverse business situations of a company, a certain department or specific case type. If the similarity between the current state and the feature of a cluster exceeds a certain threshold, knowledge workers are suggested with that ad hoc action. If there are multiple pattern matches found in the UTA knowledge, then the recommendation function provides a score from 1-5 indicating the quality of the recommendation (i.e., the higher the score, the better).

III. INTEGRATING UTA INTO ACM

Fig. 2 displays the configuration of the UTA in the ISIS Papyrus ACM Solution. The UTA is represented by an UTA object that is attached to every business case in which the UTA should collect knowledge. In Fig.2 we show the 'UTA Training Case' and its elements marked by numbers for further referencing. Apart from the basic ACM elements of a case (2.1), which are goal, artifacts, processes, tasks, etc., this case includes the UTA object (2.2) containing the parameters applied for the UTA learning process. There are several crucial

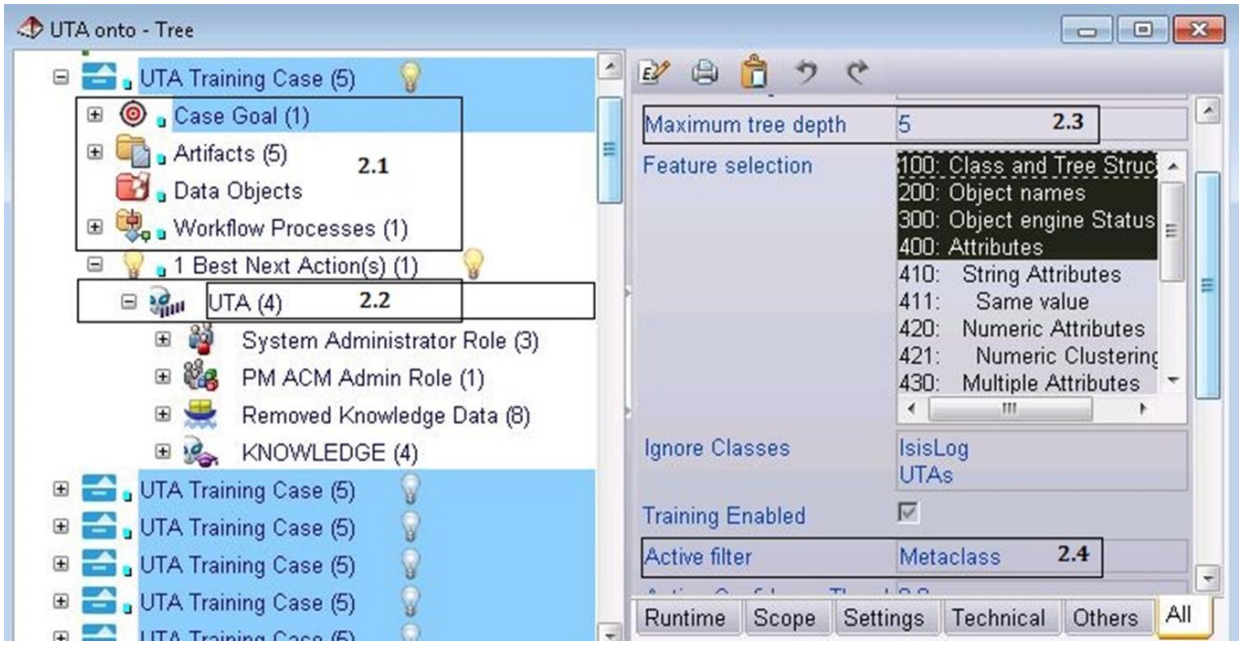

Fig. 2. The UTA configuration

parameters that influence the learning speed and the result confidence of the UTA. The 'Maximum Tree Depth' property (2.3) defines the depth up to which the case trees are observed by the UTA. Objects that are deeper in that tree will not be considered. The 'Active Filter' property (2.4) defines a Class filter for derived metadata definitions in the Papyrus Repository that are considered for the state space observations for learning. Case objects which are not derived from these classes are ignored by the UTA.

Fig. 3 shows the internal knowledge structure of the UTA in a typical learning process. The 'Knowledge' item (3.1) consists of the 'Training Sample Container', 'Action Container', 'Feature Container' and 'Cluster Container'. The 'Training Sample Container' (3.2) is a collection of all samples that are passed to the UTA for learning whenever a knowledge worker executes an ad hoc action. There are three types of learning actions:

- Positive learning is performed by normal work of knowledge workers.
- Explicit negative samples are data constellations where knowledge workers explicitly tell the UTA that a particular action is not valid, by denying the proposed BNA.
- Implicit negative learning for an action assumes that samples used for a different action are negative samples for this action. This can be configured on the UTA settings.

The 'Action Container' (3.3) contains all learned actions and is updated when a previously unknown action is observed. All information about the used parameters is captured.

The 'Feature Container' (3.4) collects all available features (data attributes) for a certain action. Each feature contains information about the Repository class, the feature type and additional details about the used categorization.

The 'Cluster Container' (3.5) contains a decision cluster for each action including references to the relevant features for the decision.

Fig. 4 shows the recommendation result of the UTA in two previously inexperienced situations for the two ad hoc actions T1 and T2, depicted in the bold and light lines, respectively. The vertical dimension represents the confidence of the UTA for a certain recommendation; the horizontal dimension represents action execution iterations by the knowledge worker. The aim of this learning is to examine the number of UTA observations needed for a specific ad hoc action and how the learning of an ad hoc action influences the other's confidence scores.

To perform analysis of the UTA functions, we can apply two types of actions as they will also happen in real work structures: positive and negative learning. Positive learning implies that for this pattern in this state space the UTA should recommend this action. In contrast, the UTA will learn not to recommend this action for negative learning where wrongly recommended BNAs are marked as incorrect by the user performing this action. The combination of both types increases the speed of learning and the quality of the recommendation considerably.

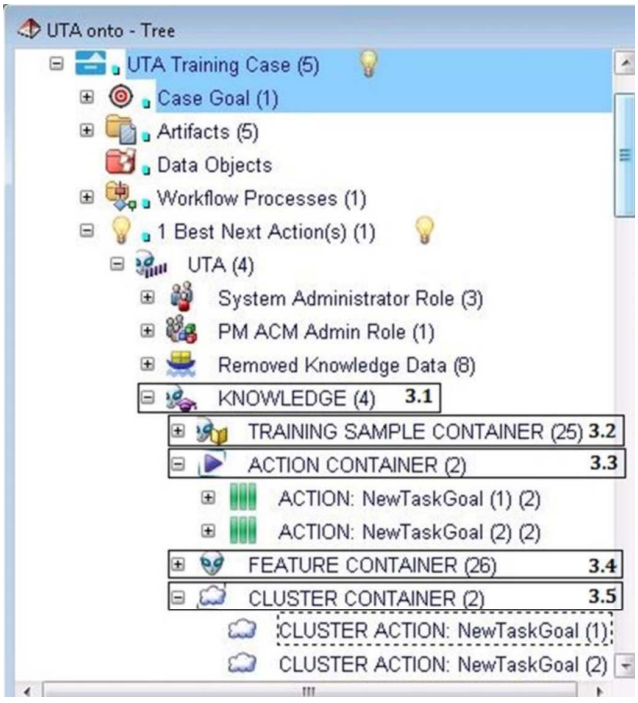

Fig. 3. The UTA knowledge

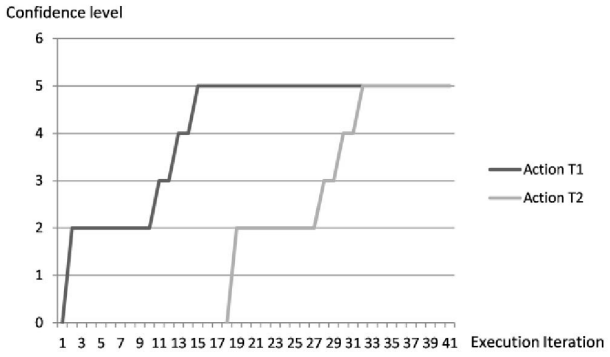

Fig. 4. The UTA training

Considering the learning of action T1, at the second observation, the recommendation is already available and the confidence level is 2. It means at the second time, when the user creates a similar state as the one from the first time and asks for the BNA, the decision sample is passed to the UTA which finds it matching with the first one and thus recommends it with still low confidence level. After exceeding the configurable number of minimum learning steps, which is 10 in this case, the confidence level starts to rise rapidly and reaches, after seventeen executions, the maximum level of 5 and then stays on that best available level. This means that a continuous observation of ad hoc actions by the UTA will provide knowledge workers with satisfying BNA suggestions just after few executions.

The learning of action T2 is started at the 18th iteration, after the execution of T1. Note that the state space for T2 is different from the state of T1. Therefore, at the 19th iteration, the confidence for T2 starts to increase while the confidence of T1 stays at 5. The learning curve of T2 is similar to the one from T1, regardless of the existence of T1. Such a behavior gives evidence that the learning of these two ad hoc actions is independent of each other.

The UTA is fully integrated into the ISIS Papyrus ACM Solution [4,5] to support the BNA as seen in Fig. 5. Knowledge workers can call this function any time when they need decision support for ad hoc actions. Here, knowledge workers are suggested to add a task to the current goal called ‚Compose an Offer'. The confidence level is at its highest

Fig. 5. The 'Best next actions' function in the ISIS Papyrus ACM system

indicated by 5 stars. If it is not an expected action in the current state, knowledge workers can reject the suggestion by clicking the x round button on the right side. In this case, an explicit negative training sample is sent to the UTA.

IV. Related Work

The work presented in this paper relates to the general stream of research on giving recommendations in process-aware information systems. The work by Koschmider et al. [14] provides recommendations to process modelers based on Lucene score. Recommendation concepts have been designed for supporting process modeling also based on action patterns [15] and on activity neighborhood [16]. The recommendation of activities at run time is mostly discussed from the perspective of process history and past execution, e.g. [17, 18], which has its roots in the adaptation of case-based reasoning concepts to workflow management.

The work presented here also adopts concepts from ontology matching [19]. Recently, this area of research has been extended towards matching of process models. Typically, works in this area integrate concepts of behavioral similarity [20] into the matching procedure. Early works like the ICoP framework [21] have been extended with semantic concepts towards better precision and recall [22, 23]. The current state of the art is summarized in [24].

The contribution of this paper is that it is among the first to adopt ontology matching towards operational support in ACM.

V. Conclusion

In this paper we introduce an approach for ACM systems to share the knowledge between groups of knowledge workers being experts in their specific domains. The knowledge is acquired from continuous observations of ad hoc actions taken by the knowledge workers. Thus, the system transfers the experience of a particular knowledge worker to the whole team taken from observations which were previously unknown to the system and also to the other knowledge workers. The UTA is fully integrated into the ISIS Papyrus ACM Solution and its associated business solutions which invoke the UTA by observing business people who act as process performers. Business users apply their knowledge to perform actions in order to achieve well-defined goals. Actions are represented by the state space of the underlying object model. The UTA can therefore match actions of the knowledge workers in relationship to achieving these goals. It is important to point out that there is no need for an explicit training of the UTA by experts. The UTA learning is part of normal business work executed by business users acting as knowledge workers. The UTA is permanently in 'learning mode' and thus observes the performer actions in real-time.

The results derived from the presented case show that with a suitable setup, the UTA is able to support the knowledge workers efficiently with best next action recommendations. Moreover, it can be analyzed if the recommendation of a certain ad hoc action is influenced by others. The results indicate the benefits of using an UTA in an ACM system to enhance the flexibility and adaptability during run time. It leads

to an organic evolution of the system based on the knowledge observed from all knowledge workers working with the system without tedious and error-prone manual process analysis and elaborate definitions by process modeling experts.

Future work will deal with further interesting investigations about the integration of the UTA into an ACM system. The confidence ranking of BNAs could include a quality feedback from the knowledge worker. Another aspect is to base the distinction between space states not only on the explicit attribute values accessible from the underlying data model but also on implicit information contained in text contents. Therefore, semantic reasoning on the overall input data space could provide additional information for the UTA. We are also considering an ontology based approach to allow for a business domain oriented configuration of the UTA.

ACKNOWLEDGMENT

We are grateful for the testing performance by E. Weiss in cooperation between ISIS Papyrus Europe AG and University of Applied Sciences Burgenland under the supervision of Dr. Kobler.

REFERENCES

[1] M.J. Pucher, “The Strategic Business Benefits of Adaptive Case Management,” in How Knowledge Workers Get Things Done - Real-World Adaptive Case Management, L. Fischer, Ed. Florida: Future Strategies Inc, 2012, pp. 19-37.

[2] K. Swenson, Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done, Florida: Megan-Kiffer Press, 2010.

[3] M.J. Pucher. (2014, April 17). The Forrester Research Wave on DCM 2014, Available: http://acmisis.wordpress.com/2014/04/17/the-forrester-research-wave-on-dcm-2014/

[4] T.T.K. Tran, M.J Pucher, J. Mendling, C. Ruhsam, “Setup and Maintenance Factors of ACM Systems,” in OTM Workshops, Lecture Notes in Computer Science, Springer, vol 8186, pp. 172-177, 2013.

[5] H.F. Sem, T.B. Pettersen, S. Carlsen, G. J. Coll, “Patterns Boosting Adaptivity in ACM,” in OTM Workshops, Lecture Notes in Computer Science, Springer, vol 8186, pp. 102-111, 2013.

[6] P.F. Drucker, Management Challenges for the 21st Century, Harper Collins Publishers, 2001.

[7] M.J. Pucher, “Method for training a system to specifically react on a specific input,” U.S. Patent USPTO Application #20080114710, May 15, 2008.

[8] M.J. Pucher. (2010, October 1). Process Mining versus User-Trained Agent. Available: http://acmisis.wordpress.com/2010/10/01/process-mining-versus-user-trained-agent/.

[9] M.J. Pucher. (2010, May 7). The User-Trained Agent has an EYE on Goals. Available: http://isispapyrus.wordpress.com/2010/05/07/with-an-eye-on-goals/

[10] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[11] A. Bögl, M. Kobler, M. Schrefl, “Knowledge Acquisition from EPC Models for Extraction of Process Patterns in Engineering Domains,” in Proceedings der Multikonferenz Wirtschaftsinformatik, Gito-Verlag, pp. 155-171, 2008.

[12] I. Rychkova, “Towards Automated Support for Case Management Processes with Declarative Configurable Specifications,” in BPM Workshops, Lecture Notes in Business Information Processing, Springer, vol 132, pp. 65-76, 2013.

[13] S. Huber, A. Hauptmann, M. Lederer, M. Kurz, “Managing Complexity in Adaptive Case Management” in S-BPM ONE Running Solutions: 5th International Conference Proc, Fischer, H., Schneeberger, J. (Eds.), Deggendorf, Germany, Springer, Berlin, pp. 209-226, 2013.

[14] K. Agnes, T. Hornung, A. Oberweis, "Recommendation-based editor for business process modeling," in Data & Knowledge Engineering, vol 70.6, pp. 483-503, 2011.

[15] S. Smirnov, M. Weidlich, J. Mendling, M. Weske, “Action patterns in business process model repositories,” in Computers in Industry, vol 63.2, pp. 98-111, 2012.

[16] C. Nguyen Ngoc, W. Gaaloul, S. Tata, "Assisting business process design by activity neighborhood context matching," in Service-Oriented Computing, Springer Berlin Heidelberg, pp. 541-549, 2012.

[17] H. Schonenberg, B. Weber, B. F. van Dongen, W. M. P. van der Aalst, “Supporting Flexible Processes through Recommendations Based on History,” in BPM 2008 Proc, pp. 51-66, 2008.

[18] B. Weber, W. Wild, R. Breu, “CBRFlow: Enabling Adaptive Workflow Management Through Conversational Case-Based Reasoning,” in Advances in Case-Based Reasoning, Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol 3155, pp. 434-448, 2004.

[19] J. Euzenat, S. Pavel, "Classifications of ontology matching techniques," in Ontology Matching, Springer Berlin Heidelberg, pp. 73-84, 2013.

[20] R. M. Dijkman, M. Dumas, B. F. van Dongen, R. Käärik, J. Mendling, “Similarity of business process models: Metrics and evaluation,” in Information System, Semantic Integration of Data, Multimedia, and Services, vol 36.2, pp. 498-516, 2011.

[21] M. Weidlich, R. M. Dijkman, J. Mendling, “The ICoP Framework: Identification of Correspondences between Process Models,” in CAiSE Proc, pp. 483-498, 2010.

[22] H. Leopold, M. Niepert, M. Weidlich, J. Mendling, R. M. Dijkman, “Heiner Stuckenschmidt: Probabilistic Optimization of Semantic Process Model Matching, ” in BPM 2012 Proc, pp. 319-334, 2012.

[23] C. Klinkmüller, I. Weber, J. Mendling, H. Leopold, A. Ludwig, “Increasing Recall of Process Model Matching by Improved Activity Label Matching,” in BPM 2013 Proc, pp. 211-218, 2013.

[24] U. Cayoglu et al., “The Process Model Matching Contest 2013,” in 4th International Workshop on Process Model Collections:Management and Reuse, PMC-MR 2013, China, 2013.