Supporting the Optimized Execution of Business Processes through Recommendations

Irene Barba[1], Barbara Weber[2], and Carmelo Del Valle[1]

[1] Departamento de Lenguajes y Sistemas Informáticos, University of Seville, Spain
{irenebr,carmelo}@us.es
[2] Department of Computer Science, University of Innsbruck, Austria
barbara.weber@uibk.ac.at

Abstract. In order to be able to flexibly adjust a company's business processes (BPs) there is an increasing interest in flexible Process-Aware Information Systems (PAISs). This increasing flexibility, however, typically implies decreased user guidance by the PAIS and thus poses additional challenges to its users. This work proposes a recommendation system which assists users during process execution to optimize performance goals of the processes. The recommendation system is based on a constraint-based approach for planning and scheduling the BP activities and considers both the control-flow and the resource perspective.

Keywords: Flexible Process-Aware Information System, Declarative Business Processes, Recommendations, Resource allocation, Prediction.

1 Introduction

Nowadays, flexible Process-Aware Information Systems (PAISs) are required to allow companies to rapidly adjust their business processes (BPs) to changes in the environment [10]. The specification of process properties in a declarative way is an important step towards the flexible management of PAISs [3]. Due to their flexible nature, frequently several ways to execute declarative process models exist. Typically, given a certain partial trace (reflecting the current state of the process instances), users can choose from several enabled activities (i.e., activities whose execution does not violate any constraint or only lead to temporary violations [6]) which activity to execute next. This selection, however, can be quite challenging since performance goals of the process (e.g., minimization of overall completion time) should be considered, and users often do not have an understanding of the overall process. Moreover, optimization of performance goals requires that resource capacities are considered. Therefore, recommendation support is needed during BP execution, especially for inexperienced users.

The need for user assistance during the execution of declarative BPs has been picked up in previous work [9,5]. Existing proposals, however, only consider the control-flow perspective for obtaining recommendations, but not resources.

In order to address this gap and to support users of flexible PAISs during process execution in optimizing performance goals like minimizing the overall completion time (i.e., time needed to complete all process instances which were planned for a certain period), we propose the generation of optimized enactment plans. For this, activities to be executed have to be selected and ordered (planning problem [4]) considering both control-flow and resource constraints (scheduling problem [2]) imposed by the declarative specification.

For planning and scheduling (P&S) the activities in a way that the process goal is optimized, a constraint-based approach is proposed since constraint programming [7] supplies a suitable framework for modeling and solving problems involving P&S [8]. For this, the declarative model is complemented with information related to estimates regarding the number of instances, activity durations, and resource availabilities. Recommendations on possible next steps are then generated considering the partial trace and the optimized plans. Replanning is supported if actual traces deviate from the optimized plans (e.g., because estimates turned out to be inaccurate).

This paper is organized as follows: Section 2 includes an overview of our proposal, Section 3 shows the application of the proposed approach to a running example, and finally, Section 4 includes conclusions and future work.

2 Method for Generating Recommendations

To optimize the overall process performance goals, users of flexible PAISs are supported during BP execution through recommendations. A recommendation is composed by one or more enabled activities (i.e., activities which are allowed to be executed given a declarative process model and a partial trace) to be executed next, together with their resource allocations. Our proposal is based on applying optimization techniques during both build and run-time (cf. Fig. 1).

Build-time. The build-time phase focusses on the generation of optimized enactment plans from declarative BP specifications by P&S the activities.

(1) **Create Declarative Specification.** In a first step, a declarative specification covering both the control-flow and the resource perspective of the BP to be supported is created. We use ConDec [6,11], a declarative language which proposes an open set of constraints for the high-level templates between BP activities (i.e., existence, relation and negation constraints).
(2) **Extend Declarative Specification.** In order to P&S the BP activities, the declarative specification is extended by considering the estimated values for: (i) the duration of the BP activities, (ii) the number of instances executed per planning period, and (iii) resource availabilities.
(3) **Generate Optimized Enactment Plans.** Optimized enactment plans are generated by applying AI techniques for P&S the BP activities, considering the extended declarative specification. In this work, CP is selected for the generation of the optimized plans since it supplies a suitable framework for modeling and solving problems involving P&S [8] (for details see [1]).

The generated plans contain information about the number of times each BP activity is executed, the start and the completion times for each activity execution, and the resource which is used for each activity execution.

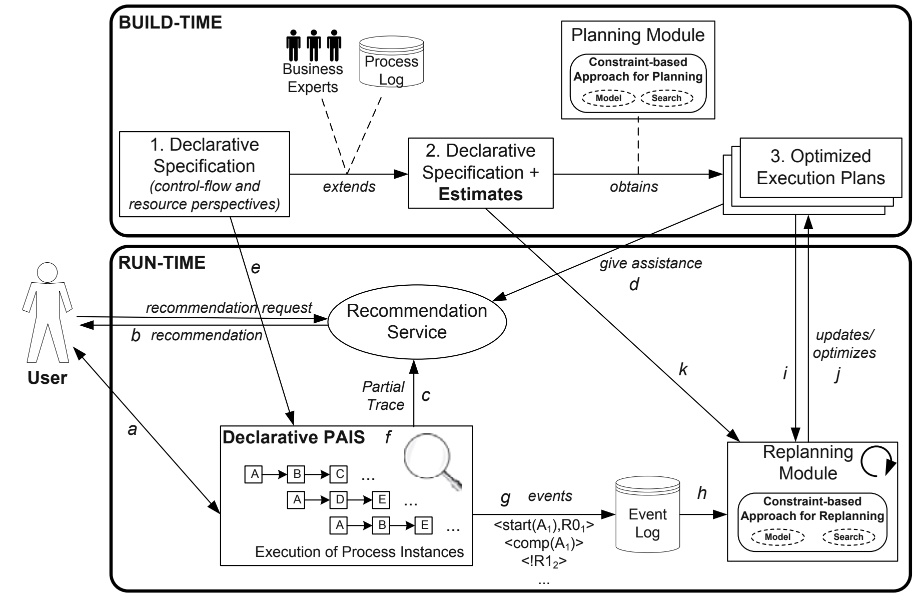


Fig. 1. Overview of our proposal

Since the generation of optimized plans presents NP-complexity, it is not possible to ensure the optimality of the generated plans for all cases. The developed constraint-based approach [1], however, allows solving the considered problems in an efficient way. Despite the NP-complexity of the considered problems, a first feasible solution can be swiftly found by a greedy algorithm.

Run-time. The plans generated in build-time are then used for giving recommendations at run-time. At run-time, process instances are executed by authorized users (`a` in Fig. 1). At any point during the execution of a process instance, the user can select from the set of enabled activities what to do next. However, to guide the user to optimize the overall process goals, recommendations are provided by the recommendation service (`b` in Fig. 1), i.e., proposing the most suitable activity to execute next[1]. For this, the recommendation service considers the current partial traces of the process instances (`c` in Fig. 1) and the best available enactment plan (`d` in Fig. 1) meeting the constraints imposed by the declarative specification (`e` in Fig. 1).

As execution proceeds, the enactment of the BP and the resource availabilities are monitored (`f` in Fig. 1). In particular, information regarding start and completion times of the executed activities, together with the resource availabilities are stored in the event log (`g` in Fig. 1). This information is analyzed by the Replanning Module (`h` in Fig. 1) together with the optimized plans (`i` in Fig. 1) to check if plan updates are required due to unexpected events. The

[1] For the current work, the durations of the recommendation request and the response time are considered negligible compared to the duration of the process activities.

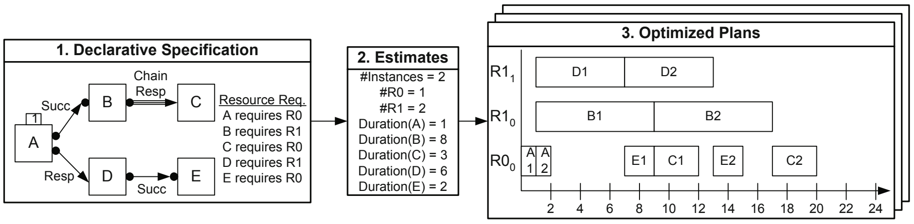


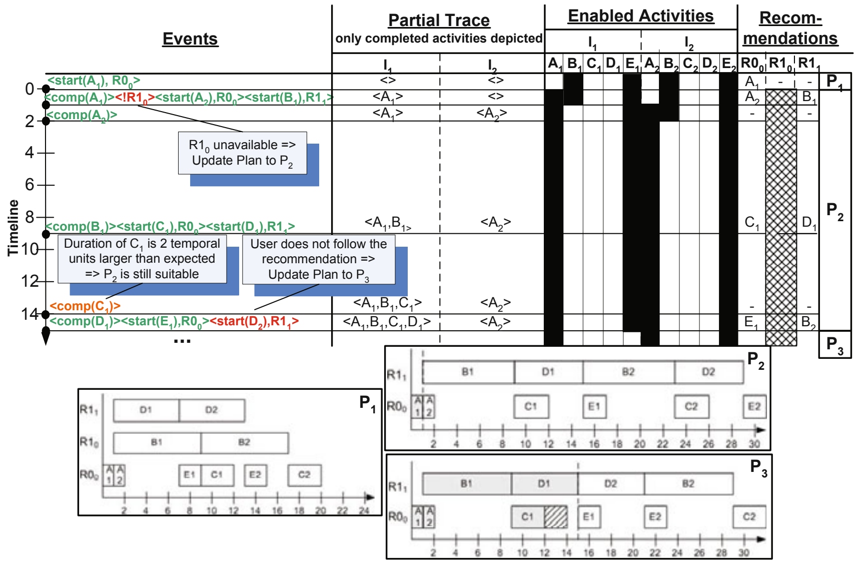
Fig. 2. Build-time for the Running Example

Replanning Module is in charge of updating the optimized plans (j in Fig. 1) in two situations: (1) there are some deviations, i.e., the execution trace is not part of one of the optimized plans (e.g., the user is not always following the recommendations) or estimates are incorrect; and (2) the Replanning Module finds a solution which is better than the current optimized plans, since this module is continuously searching for a better plan by considering the event log during BP execution, provided that the current plan is not optimal. If plan updates are required, the Replanning Module needs to access the extended declarative specification (k in Fig. 1) to generate new optimized plans. In general, despite the NP-complexity of the considered problems, replanning is less time consuming than initial planning, since most of the information about previous generated plans can usually be reused, and CSP variable values become known as execution proceeds.

3 A Running Example

In this section, our approach is used for giving recommendations during a hypothetical execution of a running example. Figure 3 shows the build-time phase for the example. The declarative specification includes 5 activities, A, B, C, D and E, and the following relations (ConDec templates [11]) between the activities (Fig. 2(1)): `Exactly_1(A)`, i.e., activity A must be executed exactly once; `Succession(A, B)`, i.e., to execute activity B, activity A needs to be executed before, and activity A must eventually be followed by activity B; `Chain Response(B, C)`, i.e, immediately after the execution of B, C must be executed; `Response(A, D)`, i.e, eventually after the execution of A, D must be executed; and `Succession(D, E)`, i.e., to execute activity E, activity D needs to be executed before, and activity D must eventually be followed by activity E. For the considered example, resources of two kinds of roles, R0 and R1, are considered. For each BP activity (Fig. 2(1)), a role is defined. In a next step, the declarative specification is extended with estimates (Fig. 2(2)). Lastly, the constraint-based approach is applied to generate optimized enactment plans for the specified problem (Fig. 2(3)). Hereby, label RI_j represents the j-th resource with role i, and label Act_k represents the k-th execution of activity Act.

Figure 3 shows the behavior of the recommendation service when two hypothetical instances with given traces are executed for the declarative specification. At the beginning of the execution, plan P_1 (which has already been generated



Events
Partial Trace
only completed activities depicted
Enabled Activities
Recom-
mendations
Timeline
<start(A1), R00>
<comp(A1)><!R10><start(A2),R00><start(B1),R11>
<comp(A2)>
R10 unavailable => Update Plan to P2
<comp(B1)><start(C1),R00><start(D1),R11>
Duration of C1 is 2 temporal units larger than expected => P2 is still suitable
User does not follow the recommendation => Update Plan to P3
<comp(C1)>
<comp(D1)><start(E1),R00><start(D2),R11>
P1
P2
P3

Fig. 3. Run-time for the Running Example

during build-time) is considered for the recommendations. The optimized plan P_1 has been created for two process instances. Initially, the partial trace for both instances I_1 and I_2 is empty (column Partial Trace, where completed events for activity executions are depicted). Furthermore, activities A, C and D of both instances are enabled (reflected by white bars), whereas activities B and E are not enabled (reflected by black bars). Activities B_1 and B_2 are not enabled since A must be executed before executing B (`Succession(A, B)`). Similarly, activities E_1 and E_2 are not enabled since the execution of E requires a previous execution of D (`Succession(D, E)`). Considering plan P_1, starting execution of activity A_1 using resource $R0_0$ is suggested. The user follows the recommendation. Due to `Exactly_1(A)`, A_1 is not enabled anymore. At time 1, A_1 is completed, hence activity B_1 becomes enabled, and the partial trace of instance I_1 contains A_1. Furthermore, an unexpected event occurs (i.e., resource $R1_0$ became unavailable), hence plan P_1 is no longer valid, and the replanning module generates plan P_2. At time 1, based on plan P_2, starting execution of activity A_2 using resource $R0_0$ and B_1 using resource $R1_1$ is suggested. The user follows the recommendation. Due to `Exactly_1(A)`, A_2 is not enabled anymore. At time 2, A_2 is completed, hence activity B_2 becomes enabled. At time 9, B_1 is completed, and starting execution of activity C_1 using resource $R0_0$ and D_1 using resource $R1_1$ is suggested. The user follows the recommendation. At time 14, C_1 is completed two time units later than expected. Even with the occurrence of this unexpected event, plan P_2 is still valid due to the slack time between activity C_1 and activity E_1. At time 15, D_1 is completed, and activity E_1 becomes enabled. Starting execution of activity E_1 using resource $R0_0$ and B_2 using resource $R1_1$ is suggested. The user partially follows the recommendation, so that, instead of executing B_2 she starts D_2. After this unexpected decision, plan P_2 becomes invalid, and the replanning module generates plan P_3. From now on, the BP execution proceeds without deviations.

4 Conclusion and Future Work

We propose a recommendation system for giving users assistance during process execution in flexible PAISs to optimize performance goals of the processes (i.e., minimization of overall completion time). The recommendation system is based on a constraint-based approach, which is used for P&S the activities such that the process goal is optimized. In the proposed approach, both control-flow and resources are considered. Furthermore, the optimized enactment plans are updated by replanning techniques when necessary. As for future work, it is intended to extend the proposed approach by considering further objective functions.

Acknowledgments. This work has been partially funded by the Spanish Ministerio de Ciencia e Innovación (TIN2009-13714) and the European Regional Development Fund (ERDF/FEDER).

References

1. Barba, I., Del Valle, C.: A Constraint-based Approach for Planning and Scheduling Repeated Activities. In: Proc. COPLAS, pp. 55–62 (2011)
2. Brucker, P., Knust, S.: Complex Scheduling (GOR-Publications). Springer-Verlag New York, Inc., Secaucus (2006)
3. Fahland, D., Mendling, J., Reijers, H.A., Weber, B., Weidlich, M., Zugal, S.: Declarative versus Imperative Process Modeling Languages: The Issue of Maintainability. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 477–488. Springer, Heidelberg (2010)
4. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann, Amsterdam (2004)
5. Haisjackl, C., Weber, B.: User Assistance During Process Execution – An Experimental Evaluation of Recommendation Strategies. In: Proc. BPI (2010)
6. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-Based Workflow Models: Change Made Easy. In: Meersman, R. (ed.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
7. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming. Elsevier (2006)
8. Salido, M.A.: Introduction to planning, scheduling and constraint satisfaction. Journal of Intelligent Manufacturing 21(1), 1–4 (2010)
9. Schonenberg, H., Weber, B., van Dongen, B.F., van der Aalst, W.M.P.: Supporting Flexible Processes through Recommendations Based on History. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
10. van der Aalst, W.M.P., Jablonski, S.: Dealing with workflow change: identification of issues and solutions. IJCSE 15(5), 267–276 (2000)
11. van der Aalst, W.M.P., Pesic, M.: Specifying, discovering, and monitoring service flows: Making web services process-aware. In: Technical Report BPM-06-09, BPM-center.org (2006)