Keywords: Planning, Scheduling, Business Processes Management.

Abstract: Usually, business process models are manually achieved by business analysts and most of current modelling languages are of imperative nature. As a consequence, non-optimized or faulty models can be obtained. This work proposes a planning based approach to give business analysts assistance for the process models generation. This approach entails the selection and the order of the activities to be executed (planning), and the resources allocation involving temporal reasoning (scheduling), both considering function optimization. The process information is specified in a declarative way, that is translated into the standard planning language PDDL. A friendly graphic language is used (ConDec-R, an extension of ConDec).

1 INTRODUCTION

In the past few years, most of the organizations need to adapt to the new commercial conditions as well as to respond to competitive pressures, so there exists an increasing interest in the effective management of business processes (BP). BP Management (BPM) supports BP using methods, techniques, and software to design, enact, control and analyze processes involving humans, organizations, documents and other sources of information (van der Aalst et al., 2003). Scheduling problems (Brucker and Knust, 2006) entails the suitable generation of execution plans for a set of tasks related by temporal and resource constraints, optimizing some functions. In a wider perspective, in Artificial Intelligence (AI) planning (Ghallab et al., 2004), the tasks to be executed are not established a priori, so it is necessary to select a suitable set of actions that must be executed in a correct order, generally optimizing some objectives. In the past years, there is an increasing interest in the application of AI P&S techniques to automate the production and execution of BP (Kearney et al., 2003; González-Ferrer et al., 2009; Barba and Del Valle, 2010).

In BPM systems, in general, a user specifies the model through a modelling language, such as BPMN (White and et al., 2004). In order to design a suitable model, the user must deal with several aspects, such as the resource allocation, the tasks properties or the relations between them, possibly optimizing some functions. In most cases, the BP information is provided to the system through imperative modelling languages. In this work, it is proposed a declarative language for the BP information. The work (Fahland et al., 2010) analyzes the differences between imperative and declarative process modelling languages with respect to build-time modifications (maintainability).

Several works concerning BP based on Linear Temporal Logic, LTL (Clarke Jr. et al., 1999), can be found. (Pesci and van der Aalst, 2006; van der Aalst and Pesci, 2006) propose a graphic tool for modelling the processes through some templates that can be translated to LTL formulas. ConDec (Pesci and van der Aalst, 2006) is a declarative language to specify dynamic BP models using a graphical notation which can be mapped to formulas in LTL. In this work, an extension of ConDec, named ConDec-R (Sect. 2), has been defined since ConDec does not allow reasoning about resources directly.

The Planning Domain Definition Language (PDDL) (Ghallab and et al., 1998) is a (standard) language for the specification of planning problems and solutions, so that any generic planner that supports PDDL is capable to solve a wide scope of problems of different nature specified through this language. PDDL 2.2 (Hoffmann and Edelkamp, 2005) also allows handling of numeric values, durative actions, plan objective functions, derived predicates and timed initial literals.

In Fig. 1, a graphical representation of the current
A PLANNING AND SCHEDULING PERSPECTIVE FOR DESIGNING BUSINESS PROCESSES FROM DECLARATIVE SPECIFICATIONS

2 DECLARATIVE SPECIFICATION OF BP

In this work, the user must provide the process information to the system in a declarative way. In order to do this, an extension of ConDec (ConDec-R) is used, including information about the resources required for the execution of the activities. ConDec (Pesic and van der Aalst, 2006) is a graphical language based on declarative specifications for modelling and enacting dynamic BP. For the definition of relationships between activities, ConDec proposes an open set of constraints based on LTL. One important difference when modelling with ConDec is that a ConDec-activity represents multiple executions of a P&S-activity, so that a ConDec-activity can be executed several times.

The main contribution of ConDec-R regarding to ConDec, is the reasoning about resources. Unlike ConDec, in ConDec-R the activities execution requires resources of a specific role, and there are several resources with the competences defined by a role. This information can be easily added to ConDec. A ConDec-R problem specification must include (an example can be seen in Sect. 4):

- As in ConDec, the tasks that can be executed in the BP enactment. In some cases, constraints about the number of times that one activity must be executed are specified (it appears above the associated activity in the graphical representation).
- The role of the required resource and the estimated duration of each task, which are specified on the left side of the task (extension of ConDec).
- Available resources with the competences of a role (extension of ConDec).
- As in ConDec, the relations between activities through constraint templates (parameterized graphical representations of LTL formulas).

3 TRANSFORMATION FROM CONDEC-R TO PDDL 2.2

PDDL (Ghallab and et al., 1998) is a (standard) language for the specification of planning problems and solutions, so that any generic planner that supports PDDL is capable to solve a wide scope of problems.
of different nature specified through this language. PDDL 2.2 (Hoffmann and Edelkamp, 2005) specifications include a domain file and a problem file.

3.1 Domain Description

A PDDL 2.2 domain contains the following items:

- **Predicates**: They represent the properties of objects that can be true or false. Several predicates have been considered for the BP execution plan generation (Table 1). Different predicates for the temporal locks of the activities are used in order to differentiate between the reasons of the lock.

- **Functions (Fluents)**: They represent values, that can vary over time, associated to objects, allowing handling of numeric values. We have used two kind of fluents: (duration ?act), that is the constant duration of the activity act; and (n-times-activity) that represents the number of times that an activity has been executed until the current state.

- **Actions/Operators (Durative)**: They allow the evolution of the system by means of state changes. For the BP execution plan generation, there exists one durative action representing the execution of each activity. In the proposed approach, there exists a PDDL 2.2 durative action associated to each ConDec-R activity, since both are durative and, also, can be executed several times. Furthermore, all the actions have a base specification (Fig. 2) that is extended through preconditions and effects depending on the relations in which the concerning activity is involved (Sect. 3.3). For the execution of an activity (Fig. 2), two conditions (condition) must be satisfied: it must not be locked for any reason, and there must exist a free resource with the role required by the activity. The predicates forced and locked are used in order to ensure the feasibility of the constraints established by the ConDec-R templates. After the activity execution, some effects (effect) are given: the n-times function is increased; the activity becomes not forced to execute; and the required resource becomes busy (only) during the activity execution.

3.2 Problem Description

A PDDL 2.2 problem contains the following items:

- **Objects**: They represent the things in the world that are noteworthy for the specified problem. In the current proposal, three kinds of objects can be distinguished: activity, resource and role.

- **Initial State**: Initially, all the resources are considered free and the number of times one activity has been executed is 0 (fluent (n-times-activity) ). Also, it is necessary to include the predicates (role

3.3 Transformation from ConDec-R Templates to PDDL 2.2

ConDec-R considers the same templates than ConDec (van der Aalst and Pesic, 2006). For a specific problem, the relations between the activities can extend the base durative action specification (Fig. 2); and the base goal for the problem specification. As follows, the considered relations are described, together with the effect they have in the PDDL 2.2 specification:

I) Existence constraints:

1. **EXISTENCE_N(A)**: A must be executed more or equal than N times. The goal (>= (n-times-A) N) is added.

2. **ABSENCE_N(A)**: A must be executed less than N times. The goal (< (n-times-A) N) is added.

3. **EXACTLY_N(A)**: A must be executed exactly N times. The goal (= (n-times-A) N) is added.

II) Relation constraints:

1. **RESPONDED EXISTENCE(A,B)**: If A is executed, then B also must be execu-
A PLANNING AND SCHEDULING PERSPECTIVE FOR DESIGNING BUSINESS PROCESSES FROM DECLARATIVE SPECIFICATIONS

Table 1: Predicates for the automatic generation of optimized BP execution plans.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(forced ?act - activity)</td>
<td>The activity ?act has to be executed before the end of the plan.</td>
</tr>
<tr>
<td>(locked-temp ?act - activity)</td>
<td>Act is temporarily locked to be executed due to alternate and responded absence relations (Sect. 3.3).</td>
</tr>
<tr>
<td>(locked-chain ?act - activity)</td>
<td>Act is temporarily locked to be executed due to chain relations (Sect. 3.3).</td>
</tr>
<tr>
<td>(locked-perm ?act - activity)</td>
<td>Act can not be executed anymore.</td>
</tr>
<tr>
<td>(resources ?act - activity ?ro - role)</td>
<td>Derived predicate that is defined as the disjunction of locked-temp, chain and perm.</td>
</tr>
<tr>
<td>(role ?r - resource ?ro - role)</td>
<td>Act requires a resource with the role ?ro to be executed.</td>
</tr>
<tr>
<td>(free ?r - resource)</td>
<td>There exists a resource ?r that presents the role ?ro.</td>
</tr>
<tr>
<td>(done ?act - activity)</td>
<td>The activity ?act has been executed.</td>
</tr>
</tbody>
</table>

1. EXISTENCE(A, B): If A exists, then B can not be executed. It leads to add a condition to all the activities different from B (even B): (at start (locked-chain B)).

2. CO-EXISTENCE(A, B): The execution of A forces the execution of B, and vice versa. The goal (= (> (n-times-A) 0) (> (n-times-B) 0)) is added.

3. ALTERNATE RESPONSE(A, B): After the execution of A, B must be executed always. It leads to add an effect to the durative-action associated to A: (at end (forced B)).

4. CO-RESPONSE(A, B): Before A, B must have been executed. It leads to add a condition to the action associated to B: (at start (done A)).

5. SUCCESSION(A, B): Relations Alternate Response(A, B) and Precedence(A, B) must hold.

6. ALTERNATE PRECEDENCE(A, B): After the execution of A, B must be executed, and between each two executions of A, there must be at least one execution of B. It leads to add two effects to A: (at end (forced B)) and (at start (locked-temp A)); and an effect to B: (at end (not (locked-temp B))).

7. ALTERNATE PRECEDENCE(A, B): Before the execution of B, A must have been executed, and between each two executions of B, A must be executed. It leads to add: a condition to B: (at start (done A)); an effect to B: (at start (locked-temp B)); and an effect to A: (at end (not (locked-temp B))).

8. ALTERNATE SUCCESION(A, B): Relations Alternate Response(A, B) and Alternate Precedence(A, B) must hold.

9. CHAIN RESPONSE(A, B): Straight after A, B must be executed. It leads to add one effect to A for each activity C that does not match with B: (at end (locked-chain C)); and one to B for each activity C that does not match with A: (at end (not (locked-chain C))).

10. CHAIN PRECEDENCE(A, B): Straight before A, B must be executed. It leads to add an effect to A: (at start (locked-chain B)); and an effect to all the activities different from A even B: (at start (locked-chain B)).

11. CHAIN SUCCESION(A, B): Relations Chain Response(A, B) and Chain Precedence(A, B) must hold.

III) Negation constraints:

12. RESPONDED ABSENCE and NOT CO-EXISTENCE(A, B): If B is executed, then A can not be executed, and vice versa. It leads to add an effect to A: (at start (locked-temp B)) and one to B: (at start (locked-temp A)).

13. NEGATION RESPONSE, NEGATION PRECEDENCE, NEGATION SUCCESION(A, B): After the execution of A, B can not be executed. It leads to add an effect to A: (at start (locked-temp B)) and one to B: (at start (locked-temp A)).

14. NEGATION ALTERNATE RESPONSE(A, B): Between two executions of A, B can not be executed. It leads to add an effect to B: (at start (when (done A) (locked-temp B))) and one to A: (at start (not (locked-temp B))).

15. NEGATION ALTERNATE PRECEDENCE(A, B): Between two executions of B, A can not be executed. It leads to add an effect to A: (at start (when (done B) (locked-temp B))) and one to B: (at start (not (locked-temp B))).

16. NEGATION ALTERNATE SUCCESION(A, B): Relations Negation
Acme Travel receives an itinerary from Karla, the customer. After checking the itinerary for errors, the process of generating the itinerary continues. One extension of the ConDec-R problem is automatic execution, non-deterministic effects and optimization.

In order to overcome this problem, in this work, the algorithm 1, so generic planners can automatically treat all the features required: durative actions, simultaneous action execution, non-deterministic effects and optimization.

The set of templates can be extended (van der Aalst and Pesic, 2006). One extension is the relation \( A \rightarrow R \rightarrow B \) (branched constraints) (van der Aalst and Pesic, 2006), that was specified through DecSerFlow language in (van der Aalst and Pesic, 2006). As follows, the considered problem is generated.

The basic idea of Alg. 1 is explained as follows: each non-deterministic relation \( ndr(r,A,B,C) \), where \( r \) is the given relation, \( A \) is the source and \( B,C \) are the sinks, means that, at least, one of relations \( r(A,B) \) or \( r(A,C) \) must be given. In order to treat both possibilities, two deterministic problems are solved, one considering the deterministic relation \( r(A,B) \), and the other one considering \( r(A,C) \). It is necessary to consider both possibilities each time the source activity is executed, leading to a limitation in the ConDec-R problems to be treated with the proposed approach: the maximum cardinality of the activities that are source of a non-deterministic relation, must be specified in the ConDec-R problem in order to generate the corresponding deterministic problems in a suitable way. Let \( n \) be the number of non-deterministic relations taking into account the maximum cardinality of the activities that are source of a non-deterministic relation. Then, in general, \( 2^n \) deterministic problems can be generated in order to deal with all the possibilities. In Alg. 1, the input is a ConDec-R non-deterministic problem \( NDP < DR, NDR > \), composed by a set of deterministic relations \( DR \), and a set of non-deterministic ones \( NDR \). As a result, a set of deterministic problems \( DP < DR \), is obtained. \( Probs \) is a set that contains \( 2^n \) deterministic problems at the end of the algorithm. The function \( DFromND(\cdot, int i, set NDR) \) is in charge of generating different combinations of deterministic relations for each problem from the set \( NDR \).

**Algorithm 1: Deterministic ConDec-R problems.**

**Function: DFromND(\cdot, int i, set NDR) .**

1. **Alternate Response** \((A, B)\) and **Negation** \(A \rightarrow R \rightarrow B\) must hold.
2. It leads to add one effect to \( B\): \( \{\text{start when (done A) \ (locked-perm A)}\} \) ; and an effect to all the activities \( A\): \( \{\text{start when (done B) \ (locked-perm B)}\} \).

**Negation Chain Response, Negation Chain Precedence, Negation Chain Succession** \((A, B)\): \( B \) can not be executed straight after the execution of \( A\). It leads to add an effect to \( A\): \( \{\text{start when (done \ (locked-chain B) \ (locked-perm A))} \} \) ; and an effect to all the activities but \( A\): \( \{\text{start (locked-chain B))} \).

The best of our knowledge, at present there is not an available planner able to treat all the features required: durative actions, simultaneous action execution, non-deterministic effects and optimization.
3. If any of the two reservation tasks fails, the itinerary is cancelled by performing the "compensate" activity and Karla is notified of the problem.

4. Acme Travel waits for confirmation of the two reservation requests.

5. Upon receipt of confirmation, Acme Travel notifies Karla of the successful completion of the process and sends her the itinerary details.

6. Once Karla is notified of either the success or failure of her requested itinerary, she may submit another travel request.

The activities in (van der Aalst and Pesic, 2006) are modelled as web services. Conversely, in this work, the activities are tasks that need to use some shared resources to be executed. Two roles are considered: Book Manager (BM) and Secretary (S). Also, it is estimated that two resources with role Book Manager (BM1 and BM2), and one with role Secretary (S), will be available for the BP enactment. Considering that only one instance is executed at the same time, the total time of the resulting plan must be minimized. In Fig. 3 the ConDec-R model of the problem is shown. As can be seen, eleven activities are presented:

**Receive Request:** A Secretary must attend the client request. This activity will be done exactly once.

**Hotel Search:** A BM asks for information to several hotel agencies to make the appropriate reservation. After the execution of Receive Request, it must be executed always, and before its execution, Receive Request must have been executed also (relation succession).

**Book Hotel:** A BM must book the appropriate reservation. Hotel Search has to be executed before it (precedence).

**Receive Failed Hotel:** A BM must receive the failure notification in the case it happens. Hotel Search has to be executed before it (precedence). On the other hand, after Hotel Search, one of the activities Book Hotel or Rec. Failed Hotel must be executed (branched response). Lastly, only one of Book Hotel and Receive Failed Hotel can be

Figure 3: ConDec-R specification for Acme Travel Company.

Figure 4: PDDL 2.2 domain specification for the Acme Travel Company.
A Secretary must study the compensation for the client. This activity has to be preceded for at least one of Receive Failed Hotel or Airline (branched precedence relation). It can not be executed after neither Hotel Search nor Airline Search (not response).

Notify Failure: A Secretary must report the failure. It has to be preceded by Compensation.

Credit Card: A Secretary proceed to make the payment. One of Book Hotel or Book Airline has to be executed before it (branched precedence). Also, after Book Hotel and Book Airline, it must be executed (response relations). One and only one of Notify Failure and Credit Card must be executed (not co-existence and mutual substitution). Also, after Credit Card, neither Hotel Search nor Airline Search can be executed (not response).

Notify Booked: A Secretary must report the information about the book to the client. After the execution of Credit Card, it must be executed always, and before its execution, Credit Card must have been executed also (succession).

5 CONCLUSIONS AND FUTURE WORK

This work proposes a PDDL 2.2 model for the optimal BP execution plan generation when specifying the process information in a declarative way, apply-
A PLANNING AND SCHEDULING PERSPECTIVE FOR DESIGNING BUSINESS PROCESSES FROM DECLARATIVE SPECIFICATIONS

Figure 7: BPMN for the Acme Travel Company.

ACKNOWLEDGEMENTS

This work has been partially funded by the Consejería de Innovación, Ciencia y Empresa de Junta de Andalucía (P08-TIC-04095) and by the Spanish Ministerio de Ciencia e Innovación (TIN2009-13714) and the European Regional Development Fund (ERDF/FEDER).

REFERENCES


