Supporting Users during the Execution of Declarative Business Process Models in Scenarios Subject to Uncertainty

Andrés Jiménez Ramírez, 48875924-G

ajramirez@us.es

Supervised by Dr. Carmelo Del Valle Sevillano and Dr. Irene Barba Rodríguez

Thesis Dissertation submitted to the
Departamento de Lenguajes y Sistemas Informáticos
at Universidad de Sevilla in fulfillment of the
requirements for the degree of Doctor of Philosophy.

(Thesis Dissertation)

Leopold-Franzens-Universität Innsbruck
Institute of Computer Science
Quality Engineering Research Group
Assoc. Prof. Priv.-Doz. Dr. Barbara Weber
Technikerstraße 21a, 2. OG
6020 Innsbruck, Austria

QUALITY ENGINEERING

TO WHOM IT MAY CONCERN

Innsbruck, June 4th 2014-06-04

Evaluation Report regarding PhD Thesis „Supporting Users during the Execution of Declarative Business Process Models in Scenarios Subject to Uncertainty" by Andrés Jiménez Ramírez, University of Seville

The thesis of Andrés Jiménez Ramírez addresses a relevant and at the same time challenging topic in the area of business process management. Main goal of the thesis is to facilitate (1) the design of business process models and (2) to support users during run-time executing these models. To address (1) the thesis suggests an approach to automatically generate optimized enactment plans from a declarative specification. For managing the generated plans configurable process models are used. To address (2) the thesis proposes an approach for configuring the generated configurable process models at run-time, i.e., to select one optimized execution plan in a step-wise manner.

The contributions made by Andrés Jiménez Ramírez are technically sound, original, thoroughly evaluated, and extend the state-of-the art considerably. Regarding (1) the thesis extends the existing state-of–the art by considering more expressive declarative specifications (i.e., temporal, data, and resource constraints; stochastic attributes), by considering multi-objective optimization and by addressing uncertainty through flexibility and robustness. Regarding (2) the thesis goes beyond the existing state-of-the art by enabling run-time configuration through a completely automated approach based on questionnaires. The research described in this thesis has also led to several high quality publications covering all core parts of the thesis, which underlines the quality of the research. Especially, the CAiSE 2013 paper (CORE A and acceptance rate < 15%), which describes the generation of optimized plans considering multi-objective functions, stands out (improving the IJCIS 2013 article, in which the author has been involved, in several important aspects). Also the second main part of the thesis regarding run-time configuration has been published well at BIS 2014 (CORE B) and ICAART 2014 (CORE C). This is complemented by a paper published in ISD 2012, describing the tool for generating optimized execution plans (CORE A). Moreover, it is worth mentioning that Andrés Jiménez Ramírez received the best paper award for the BIS 2014 paper and that a journal version covering the main parts of the thesis just got accepted for the Information and Software Technology journal (IF 1.522). I expect the thesis of Andrés Jiménez Ramírez to create considerable impact in the research community, since the proposed techniques provide novel possibilities for designing business processes and for executing them during run-time.

I have no hesitation in recommending this thesis for obtaining the International Doctorate.

Signature

Assoc. Prof. Dr. Barbara Weber

TO WHOM IT MAY CONCERN

Eindhoven, July 4th 2014

Evaluation Report and overview of the PhD Thesis "Supporting Users during the Execution of Declarative Business Process Models in Scenarios Subject to Uncertainty" as prepared by Andrés Jiménez Ramírez from the University of Seville.

I wish to attest that this thesis deals with topics which can be considered both innovative and relevant in my area of expertise, i.e. business process management. As one of its prime contributions, this thesis includes a declarative language (named SDeclare) for modelling business processes. Such a language extends the declarative language Declare by considering resources, temporal constraints, data constraints, and input uncertainty. Furthermore, a constraint-based approach is proposed for automatically generating optimized enactment plans from SDeclare models, which pursue a variety of objective functions. What is more, the flexible execution of the models which are created using such a language is supported as well. To this end, an imperative configurable process model is created from the SDeclare model. That model can be configured at run-time through a questionnaire-based approach, which automatically generates questions when a decision needs to be made.

I find it impressive that all the contributions of the thesis by Andrés Jiménez Ramírez have been validated through peer review by boards associated to high-quality academic journals and scientific conferences. Specifically, the first part of the thesis has led to 5 publications in CORE A and CORE B conferences, among which the publication during CAiSE'13 (acceptance rate below 15%). In addition, this part has led to a co-authorship for two journal papers. For the second part of the thesis, two conference papers were published, one of which was honored with the best paper award (BIS'14). Finally, a scientific paper has been accepted recently by the IST journal (impact factor of 1.522).

A final point worth mentioning is that the PhD candidate has spent a semester during the academic year 2012/2013 at my institute. During that time, our team has got to know him as an intelligent and sympathetic colleague, with whom it was a joy to collaborate.

To conclude, I have no hesitation in recommending this thesis for obtaining the International Doctorate.

Prof.dr.ir. Hajo Reijers
Eindhoven University of Technology
The Netherlands

Agradecimientos

Son muchas las personas a las que agradecer la completitud de esta tesis, pero en especial a mis directores de tesis, Carmelo e Irene, por sus buenos consejos y por el apoyo constante del que me han provisto en estos años y sin el cual no habría sido posible llegar hasta aquí. Mi más sincera gratitud a Barbara cuyo empeño y dedicación han aportado un valor incalculable a este trabajo.

A Cristina por su paciencia y confianza ciega en mi trabajo que, junto a mi familia y amigos, siempre han estado ahí con un apoyo incondicional.

Gracias también a Rafael por haberme permitido empezar esta andadura y haberme guiado en mis comienzos investigadores.

Gracias a todos de corazón.

Resumen

La calidad de los modelos de los procesos de negocio (es decir, artefactos software que capturan las relaciones entre las unidades organizacionales de un negocio) es esencial para la mejora de la gestión de los procesos de negocio. Sin embargo, dicho modelado se hace normalmente a mano. Esa tarea puede representar un gran reto para el analista y consumir bastante tiempo, sobre todo en ciertos escenarios con unos requisitos de diseño concretos (es decir, estimaciones de los atributos de las actividades, incertidumbres, relaciones entre actividades y asignación de recursos). Esta situación es aún más complicada si se añaden algunos requisitos de optimización, además de flexibilidad y robustez. Además, los modelos generados puede ser poco eficientes, contener errores y, probablemente, sean muy estrictos. Para facilitar la tarea del analista y para mejorar los modelos de proceso de negocio resultantes, en esta memoria de Tesis se describe un método software para generar planes de ejecución de manera automática en tiempo de diseño a partir de una especificación declarativa. Para gestionar estos planes, esta propuesta se basa en modelos de procesos de negocio configurables, los cuales permiten a los analistas entender qué comparten esos planes y cuáles son sus diferencias.

Antes de poder ejecutar el modelo de proceso de negocio configurable, es necesario seleccionar un modelo de proceso de negocio concreto de entre los contenidos en el modelo configurable. Esta selección la hace normalmente el analista, quien manualmente individualiza el modelo teniendo en cuenta los requisitos del negocio. Para individualizar dicho modelo, al contrario que el resto de trabajos relacionados que existen en la literatura, en esta tesis se propone un método totalmente automático para crear una herramienta software basada en cuestionarios que guíe al usuario para la individualizació de un modelo de proceso de negocio configurable en tiempo de ejecución. Así, la decisión de qué aspecto tiene el plan de ejecución se retrasa hasta el tiempo de ejecución, que es cuando hay más información disponible.

La principal diferencia de la propuesta que se presenta en esta tesis frente a otros trabajos previos es la gestiòn de considera la incertidumbre de los escenarios a través de atributos estocásticos, además de la optimización de múltiples funciones objetivos. Además, se propone un herramienta basada en cuestiona-

rios para permitir, en tiempo de ejecución, la selección de un plan de ejecución optimizado a partir de una especificación declarativa.

Abstract

The quality of business process models (i.e., software artifacts that capture the relations between the organizational units of a business) is essential for enhancing the management of business processes. However, such modelling is typically carried out manually. This can be quite challenging and be very time consuming in some real scenarios which present certain design requirements (i.e., estimated activity attributes, input uncertainty, relations between activities and resource allocation). This situation is further complicated if such requirements have to be addressed together with some optimization requirements including flexibility and robustness. Moreover, the resulting models may be non-optimized, potentially contain errors, and might be too strict. To facilitate the human work and to improve the resulting business process models, this Thesis Dissertation proposes a software-supported approach for automatically generating optimized enactment plans from declarative specifications at design-time. For managing these plans the proposed approach suggests to build upon configurable business process models (which allow analysts to understand what these plans share and what their differences are).

Before the execution of the configurable business process model, a business process model has to be selected from it. This selection is typically performed by an analyst who manually individualizes the model in order to address the business requirements. To individualize such models, unlike existing approaches, a totally automated method to create a questionnaire-based application for guiding a business expert on individualizing the configurable business process model during run-time is proposed. Therefore, the decision of how the enactment plan to be executed looks like is deferred to run-time, i.e., when more information is available.

The current Thesis Dissertation differs from existing approaches since it considers the uncertainty of the scenario through stochastic attributes, as well as the optimization of multiple objective functions. Moreover, a questionnaire-based application is suggested to enable the selection of an optimized enactment plan from a declarative specification during run-time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Generalities

A business process (BP) can be defined as a set of activities which are performed in coordination in an organization to achieve a business goal (Weske, 2007). These activities can be manual activities, other BPs, or even pieces of software. In order to support BPs, BP Management (BPM) embraces methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, and other sources of information (van der Aalst et al., 2003). Such management generally follows a strict methodology to ensure the quality of the information systems which are created.

Nowadays, there exist several software tools, namely BPM Systems (BPMSs), which are intended to support the BPM during the BPM life cycle (Weske, 2007). The traditional BPM life cycle includes four phases (Weske, 2007):

1. Process design & analysis, when a design of the BP (i.e., a BP model) is created following the requirements.

2. System configuration, when the software defined in the BP model is implemented.

3. Process enactment, when the software is executed (i.e., one or more BP instances) following the BP model.

4. Evaluation, when monitoring information or logs are analyzed to look for design improvements.

In turn, it becomes increasingly common for organizations to deal with large collections of BP models (Rosa et al., 2012). Therefore, more and more research is done related to BP collections (Dijkman et al., 2012) in which configurable BP

models (La Rosa et al., 2008) are widely used to capture families of BPs in an integrated manner allowing the users for a high variability. In such scenarios a new phase, namely *configuration & individualization*, is defined in the BPM life cycle after the process design & analysis phase (La Rosa et al., 2008). Such a new phase is in charge of selecting one BP model from the configurable BP model.

In addition, in the last years, the interest in the effective and flexible management of BPs has grown considerably (Reichert and Weber, 2012; Dijkman et al., 2012) since real scenarios are generally subject to uncertainty. In a related way, flexibility and robustness concerns have received increasing attention (de Haan et al., 2011; Golden and Powell, 2000; Gueorguiev et al., 2009; Cicerone et al., 2012), also in the field of BPM (Reichert and Weber, 2012; Schonenberg et al., 2008a).

To support the BPM lifecycle, artificial intelligence (AI) planning techniques have been successfully applied at different stages since an instance of a BP is analogous to a plan in AI. AI planning (Ghallab et al., 2004) proposes techniques to select a plan (i.e., a set of activities to execute in a specific order) to achieve a given goal. In addition, the performance of an execution plan related to a BP model can be greatly influenced by scheduling decisions (Pinedo, 2008; Brucker et al., 2006) such as the resource allocation (Shah and Ward, 2003; Karim and Arif-Uz-Zaman, 2013). Such scheduling decisions are commonly made by BPMSs during the enactment phase (i.e., run-time) by automatically assigning work (i.e., activities) to the available resources (Russell et al., 2005). In general, a planning & scheduling (P&S) problem consists of determining an enactment plan for a set of activities which are related by temporal constraints, which compete for some shared resources, and where the optimization of some objective functions is pursued. In such context, constraint programming (CP) (Rossi et al., 2006) supplies a suitable framework for modeling and solving problems involving P&S aspects (Salido, 2010).

1.2 Motivation

The quality of a BP design (i.e., of a BP model) has a great influence on all the phases of the BPM life cycle and is essential for BP improvement, which has been ranked as the number one priority for top management by the 2010 Gartner survey (Group, 2010).

In the process design & analysis phase the BP models are typically specified by hand using imperative languages like EPC or BPMN (BPMN, 2011). This way, a precise activity sequence which establishes how a given set of activities has to be performed is defined. Such a sequence typically includes temporal relations between activities or even dependencies with input data. Typically, such activities

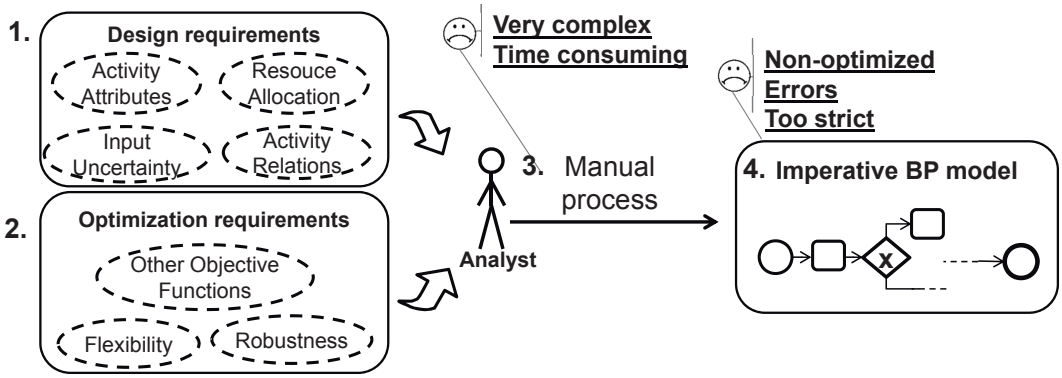

Figure 1.1: Motivation overview: Designs issues.

are related to a set of attributes (e.g., duration and cost) which need to be estimated. Furthermore, in many real scenarios such estimates might be subject to input uncertainty (e.g., normally, surgical operation durations and recovery durations are imprecisely definite) (Souki, 2011). Therefore, regarding such scenarios and motivated by the case study described in this Thesis (cf. Chapter 5), when designing a BP model, analysts have to face certain design requirements (cf. Figure 1.1 (1)), such as:

1. Dealing with activity attributes and their estimated values.

2. Managing the input uncertainty which exists in many real scenarios (Souki, 2011) in which providing a range of possible values for a BP property is most reliable that providing an exact value which may be difficult to know.

3. Dealing with relations between the activities, i.e., control-flow as well as temporal and data constraints of the BP. [1]

4. Considering resource allocation.

Since uncertain scenarios are considered, managing such input uncertainty becomes necessary. For this, flexibility and robustness are proposed since they are considered the best way to properly address the considered uncertainty. The situation is further complicated if the aforementioned design requirements have to be addressed along with optimizing some (potentially) conflicting objective functions (e.g., minimizing time and maximizing profit). Such optimization requirements (cf. Figure 1.1 (2)) can be summarized as follows:

[1] Note that the considered scenarios are focused on the control-flow and the resource perspectives of the BPs and the data perspective is only partially considered.

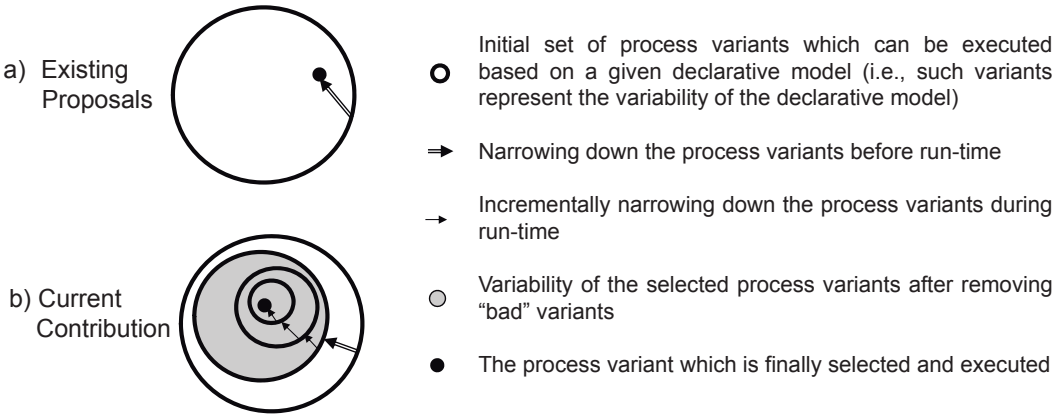

Figure 1.2: Motivation overview: Run-time issues.

1. Flexibility, i.e., the capability to adapt to input uncertainty (Reichert and Weber, 2012; Golden and Powell, 2000; Schonenberg et al., 2008a). For this, designed models should consider different execution alternatives to support such uncertain scenarios (Weske, 2007).

2. Robustness, i.e., the capability to withstand the uncertainty to some extent (Eppink, 1978; Cicerone et al., 2012; de Haan et al., 2011). For his, BP models should be designed to avoid making unnecessary adaptations which typically are costly.

3. Other objective functions are commonly considered since the BP design usually involves a trade-off between different quality dimensions which may be in conflict or be opposed (Reijers, 2003).

This task of creating a BP design can form a very complex problem and be very time consuming (cf. Figure 1.1 (3)). Moreover, the resulting models may be non-optimized, potentially contain errors, and might be too strict (Ferreira and Ferreira, 2006; Mendling et al., 2007; Westergaard and Maggi, 2012) (cf. Figure 1.1 (4)). For this, methods and tools for supporting analysts during the BP design are becoming more and more necessary.

To facilitate the human work involved in such design, to avoid failures, and to allow for a better optimization during the execution phase, declarative BP models are increasingly used since the tacit nature of human knowledge is often an obstacle to eliciting accurate BP models (Ferreira and Ferreira, 2006). However, due to their flexible nature, there are frequently several variants related to a given declarative model, each one presenting specific values for different objective functions (e.g., overall completion time or profit). Therefore, the decision about how to execute this declarative model (i.e., selecting a variant that finally gets executed) can

be quite challenging since usually many constraints need to be obeyed, multiple instances of a process get concurrently executed within a particular timeframe, shared resources need to be allocated, and relevant objective functions should be considered.

In such context, there exist some proposals for generating imperative BP models or that could be extended in such direction (cf. (Pesic, 2008; Sadiq et al., 2005; Pesic et al., 2007; Lu et al., 2009; Ferreira and Ferreira, 2006; Montali, 2009; Westergaard and Maggi, 2012; Krogt et al., 2010; Hummer et al., 2013)). These proposals are based on generating a single execution plan which fulfills all the BP constraints starting with a constraint-based specification, e.g., a declarative model. This plan could be, in turn, used for the generation of an imperative BP model. However, as a major disadvantage of existing proposals, only one single execution plan is selected (i.e., a single process variant) before starting the process execution which unnecessarily restricts the flexibility (Barba et al., 2013a) and hence diminishes the advantages of using declarative process models. In particular, if BPs are subject to uncertainty and conditions may change during BP execution, it might turn out that the selected BP model is not applicable and replanning might be required. In order to be better able to cope with such uncertainty, it is more suitable to defer the decisions of how the BP model to be executed looks like to run-time (i.e., to select the BP model to be executed incrementally during run-time). To be more specific, instead of narrowing down the selection to one single variant before run-time (cf. Figure 1.2 (a)), it would be better that only *the worst* variants are removed while the *the best* variants are kept (cf. the outermost gray circle in Figure 1.2 (b)). Thereby the goodness of a variant is measured by its values for given objective functions (Jimenez-Ramirez et al., 2013a). This way, the variants which are kept can be narrowed down incrementally during run-time at the last possible moment (i.e., gradually moving from the outermost inner circle to the back dot in Figure 1.2 (b)).

Thus, the existing proposals are not sufficient to address all the previously mentioned requirements, e.g., dealing with the flexibility needs of existing BPs (Reichert and Weber, 2012).

1.3 Contributions

In order to facilitate the human work which is involved in the process design & analysis phase and to improve the resulting imperative BP models a method for automatically creating configurable BP models (i.e, a modelling artifact that captures a family of process models in an integrated manner) (der Aalst et al., 2006) from declarative specifications (Ferreira and Ferreira, 2006) is proposed (cf. Figure 1.3). The proposed approach considers all the aforementioned requirements

which have to be considered when creating a suitable BP model, i.e., activity attributes, resource allocation, input uncertainty, relation between activities, optimization of several objective functions, as well as flexibility and robustness issues, and is detailed in the following.

Declarative models are typically easier to specify and less time-consuming than imperative models in scenarios where high variability is required (Ferreira and Ferreira, 2006). Therefore, declarative specifications are used as starting point of the proposed approach. For this, the Declare language[2] (Pesic, 2008) is used as basis, since it allows the specification of BP activities together with the constraints which must be satisfied for correct BP enactment and for the goal to be achieved. Declare is extended in order to widen its design flexibility by considering stochastic values for modelling the uncertainty of the scenario (as required in the considered problems, cf. Section 1.2), resulting in the SDeclare language. To be more precise, with the proposed extension, some properties of a BP (such as activity attributes, data and temporal constraints, and resource availability) can be expressed through probabilistic mass functions instead of with fixed values. For example, the current approach allows one to specify the uncertainty about the duration of an activity by using a flat discrete range (e.g., [15-20], meaning that such an activity may last from 15 to 20 units of time with the same probability). This can be used, for example, for specifying that the arrival time of clients is uncertain due to unpunctual clients, or that the availability of some resources is subject to uncertainty. The SDeclare language is then used for the declarative specification of the BP models (cf. Figure 1.3 (1)).

Since a declarative model captures highly variable scenarios (i.e., it allows numerous possible enactment plans), it may include many execution alternatives that are not desirable for the business regarding the optimization of a set of objective functions. For this, a desirable part of the variability of a declarative model has to be extracted (cf. Chapter 3). To do this, a method for generating optimized BP enactment plans from declarative specifications is proposed to optimize the performance of a process by considering multiple objectives (cf. Figure 1.3 (2)). This process is done automatically using a constraint-based approach which obtains the best execution alternatives of a declarative model according to a set of given objective functions. For this, activities to be executed have to be selected and ordered (planning problem (Ghallab et al., 2004)) considering both control-flow constraints as well as resource constraints imposed by the declarative specification (scheduling problem (Brucker and Knust, 2006))

Since the generated set of multi-objective optimized enactment plans may contain similar alternatives or non-robust alternatives, such set must be filtered

[2]Declare is one of the most referenced and used declarative BP languages in the context of BPM.

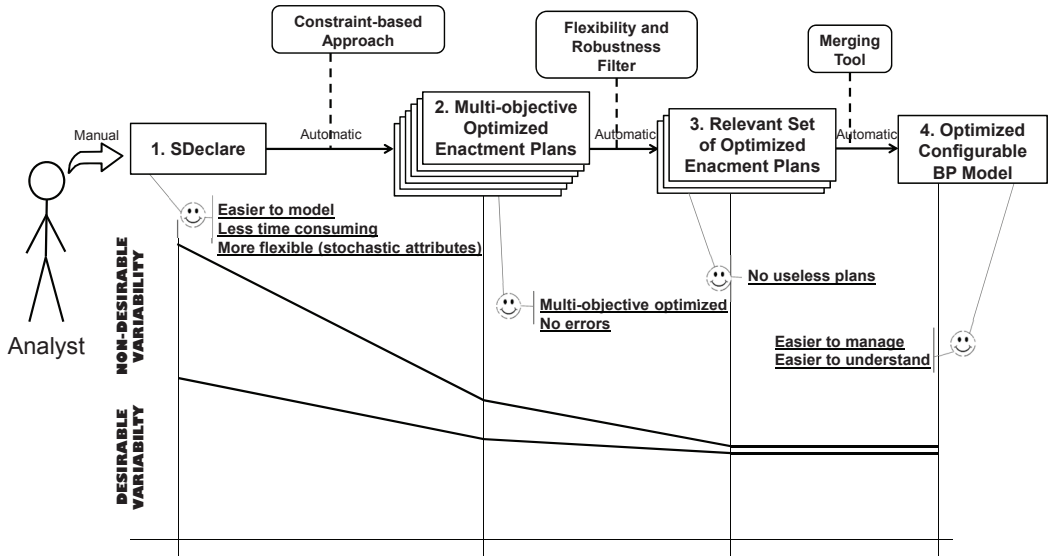

Figure 1.3: Contribution overview: Selecting the desirable variability of a declarative model.

considering flexibility and robustness concerns (cf. Chapter 4). That filter is performed regarding how the set of plans manages the input uncertainty in such a way that flexibility and robustness are optimized. Specifically, those alternatives which are too strict (i.e., not robust (Eppink, 1978; Cicerone et al., 2012; de Haan et al., 2011)) or which only withstand an extent of the uncertainty which is already withstood by another alternative (i.e., do not contribute to the flexibility of the final solution (Reichert and Weber, 2012; Golden and Powell, 2000; Schonenberg et al., 2008a)), are discarded. Therefore, the variability of the source declarative model is reduced to a set of relevant plans where most of the non-desirable alternatives are removed (cf. Figure 1.3 (3)). In this way, the proposed approach manages both flexibility and robustness at design-time[3], as motivated in Section 1.2.

Typically the relevant plans which are kept after such filtering process share many commonalities since they are created from the same declarative specification and optimize the same objective functions. For this the current approach suggests to build upon established techniques, i.e., configurable BP models (Rosemann and van der Aalst, 2007; Rosa et al., 2012; der Aalst et al., 2006; Hallerbach et al.,

[3]Note that flexibility can be managed by: (1) design, i.e., at design-time some control-flow patterns which allows one to consider different alternatives (e.g., OR structures) are included in the model or (2) flexible PAISs, i.e., at run-time several activities of a flexible BP model (e.g., a declarative model) are enabled to be executed (Reichert and Weber, 2012). Since this approach is focused on the process design & analysis phase, flexibility at run-time is out of the scope of this Thesis.

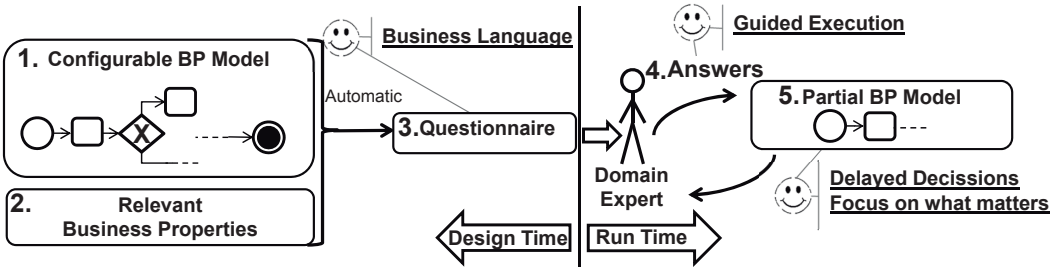

Figure 1.4: Contribution overview: Run-time individualization of configurable BP models.

2010; Sun and Aiello, 2008; Gottschalk et al., 2008). Such models can be created by merging all these plans (cf. Figure 1.3 (4)). The goal of creating configurable BP models in the current Thesis is twofold:

1. Supporting the analysts in the management of the set of optimized plans.

2. Helping the analysts understand what the different plans share, what their differences are, and why and how these differences occur (Rosemann and van der Aalst, 2007).

To enable configurable BP models being configured by domain experts in such a way that these experts incrementally reduce the number of process variants to be executed, the automatic generation of questionnaires (i.e., sequences of questions each one created for configuring a part of the related configurable BP model (La Rosa et al., 2008)) is proposed. While the usage of questionnaires for individualizing configurable BPs is not new (La Rosa et al., 2008; Rosa et al., 2009), existing works require that analysts manually create the questionnaires which configure the configurable BP model. In addition, such a configuration is done at configuration time (i.e., before process execution starts), and hence premature decision may unnecessarily be taken. To overcome such drawbacks, this Thesis Dissertation proposes a method for (cf. Figure 1.4):

1. Automatically generating the questionnaires for individualizing the configurable BP model.

2. Incrementally individualizing the configurable BP model during run-time using the automatically generated questionnaires.

For this, using the generated configurable BP model (cf. Figure 1.4 (1)) together with a set of well-defined relevant business properties (i.e., properties that can be measured within each variant and which are understandable by the domain expert, cf. Figure 1.4 (2)), a questionnaire is automatically generated without

involving the analyst. Such a questionnaire consists of questions related to the business properties written in the business language (cf. Figure 1.4 (3)). Thereafter, the domain expert interacts with the questionnaire to configure the configurable BP model herself during run-time. This way, the generated questionnaire allows to narrow down the variants of the configurable BP model in an incremental way during run-time, i.e., guiding the execution of the configurable BP model by answering the questionnaire (cf. Figure 1.4 (4)). Therefore, the BP model is partially created (cf. Figure 1.4 (5)) and thus, it is possible to execute already configured parts. In addition, as users often do not have an understanding of the overall process, they can focus only on the part of the configurable BP model to be configured, which may help them to take decisions.

Note that the proposed approach is appropriate for managing scenarios which present certain requirements, i.e., scenarios which:

1. Present high variability.

2. Pursue the optimization of some objective functions.

3. Are subject to changes (e.g., company best practices which change due to the customers feedback).

4. Have a well-defined set of business properties which can be extracted for a variant (e.g., the property 'completion time' of a variant can be related to the 'opening and closing time' of the business).

5. Highly rely on domain expert's skills (i.e., decisions influence business performance) and thus, decisions can not be predefined.

As an example of such a scenario, the suitability of the current proposal has been validated through a real scenario (cf. Chapter 5).

The main contributions of the current Thesis Dissertation are:

1. The consideration of temporal, data and resource constraints together with stochastic attributes for the declarative BP specification.

2. The management of the uncertainty of the scenario through flexibility and robustness, at the same time as the optimization of multiple objective functions is considered.

3. A questionnaire-based application to enable the selection of an optimized enactment plan from a declarative specification during run-time without involving the analyst.

1.4 Structure

The rest of the document is organized as follows:

- Chapter 2 includes background related to the areas which are addressed in the current Thesis Dissertation, i.e., (1) Planning and Scheduling, (2) Constraint Programming, (3) Business Process Management, and (4) Uncertainty Management.
- Chapter 3 details the constraint-based approach which is used for P&S the BP activities so that multi-objective optimized enactment plans are generated from a declarative specifications.
- Chapter 4 describes how the resulting set of multi-objective optimized BP enactment plans can be used to guide the optimal execution of a declarative model through automatically-generated questionnaires. In addition, both flexibility and robustness concerns are dealt to manage the input uncertainty of the scenarios.
- Chapter 5 describes a wide empirical evaluation which has been carried out to evaluate the effectiveness and the efficiency of the proposed approach.
- Chapter 6 presents a critical discussion of this Thesis Dissertation as well as its limitations.
- Chapter 7 summarizes the main conclusions which were obtained during the development of this Thesis.
- Lastly, Chapter 8 shows some future work which is intended to be addressed.

1.5 Publications

During the development of the current Thesis Dissertation, some research works have been published in different Conferences and Journals. These publications[4] support the validation of the scientific quality of this thesis.

1. *Andrés Jiménez-Ramírez, Irene Barba, Barbara Weber, Carmelo del Valle.* ”Automatic Generation of Questionnaires for Supporting Users during the Execution of Declarative Business Process Models”. 17th International

[4]The publications has been chronologically ordered starting from the most recent publications, and ending with the oldest publications.

Conference on Business Information Systems (In press) (BIS 2014, **Ranked as B in ERA and CORE Conference Rankings)**, 2014. **Awarded as Best Paper.**

2. *Andrés Jiménez-Ramírez, Barbara Weber, Irene Barba, Carmelo del Valle.* "Automatic Generation of Questionnaires for Managing Configurable BP Models". 6th International Conference on Agents and Artificial Intelligence (In press) (ICAART 2014, **Ranked as C in ERA and CORE Conference Rankings)**, 2014.

3. *Irene Barba, Barbara Weber, Carmelo Del Valle, Andrés Jiménez-Ramírez.* "User Recommendations for the Optimized Execution of Business Processes". **Data & Knowledge Engineering, Volume 86, Pages 61-84, ISSN 0169-023X**, 2013

4. *Andrés Jiménez-Ramírez, Irene Barba, Carmelo del Valle, Barbara Weber.* "Generating Multi-objective Optimized Business Process Enactment Plans". 25th International Conference (CAiSE 2013, **Ranked as A in ERA and CORE Conference Rankings**), **Springer LNCS Volume 7908, Pages 99-115** , 2013

5. *Irene Barba, Carmelo del Valle, Barbara Weber, Andrés Jiménez-Ramírez.* "Automatic Generation of Optimized Business Process Models from Constraint-based Specifications" **International Journal of Cooperative Information Systems, Volume 22, Issue 02, Pages 59, ISSN 1793-6365**, 2013

6. *Andrés Jiménez-Ramírez, Irene Barba, Carmelo del Valle, Barbara Weber.* "OptBPPlanner: Automatic Generation of Optimized Business Process Enactment Plans". 21th International Conference on Information System Development (ISD 2012, **Ranked as A in ERA and CORE Conference Rankings**), **Building Sustainable Information Systems, Pages 429-442, ISBN 978-1-4614-7539-2**, 2012

7. *Andrés Jiménez-Ramírez, Irene Barba, Carmelo del Valle, Barbara Weber.* "Generating Multi-objective Optimized Configurable Business Process Models". 6th International Conference on Research Challenges in Information Science (RCIS 2013, **Ranked as B in ERA and CORE Conference Rankings**), Pages 1-2 , 2012

8. *Andrés Jiménez-Ramírez, Rafael M. Gasca, Angel Varela-Vaca.* "Contract-based Test Generation for Data Flow of Business Processes using Constraint Programming". 5th International Conference on Research Challenges

in Information Science (RCIS 2011, **Ranked as B in ERA and CORE Conference Rankings**), Pages 1-12 , 2011

9. *Angel Varela-Vaca, Rafael M. Gasca, Andrés Jiménez-Ramírez.* "A Model-driven Engineering Approach with Diagnosis of Non-conformance of Security Objectives in Business Process Models". 5th International Conference on Research Challenges in Information Science (RCIS 2011, **Ranked as B in ERA and CORE Conference Rankings**), Pages 1-6 , 2011

1.6 Research Projects

The development of the current Thesis Dissertation has been framed in and funded by some research projects[5].

1. **Técnicas para la diagnosis, confiabilidad y optimización en los sistemas de gestión de procesos de negocio.** Ministerio de Ciencia e Innovación TIN2009-13714 *(17/04/2010 - ..)*.

2. **OPBUS: Mejora de la calidad de procesos mediante tecnologías de optimización y tolerancia a fallos.** Consejería de Innovación, Ciencia y Empresa *(17/04/2010 - 12/01/2011)*.

[5]The research projects has been chronologically ordered starting from the most recent projects, and ending with the oldest projects.

Chapter 2

Background

This Thesis Dissertation combines aspects of Planning & Scheduling (P&S), and Constraint Programming to support users during the execution of BPs. In a related way, Section 2.1 gives an overview of Planning & Scheduling, Section 2.2 describes the constraint programming paradigm. Section 2.3 provides backgrounds regarding BPM, and Section 2.4 states different mechanisms for dealing with uncertainty.

2.1 Planning and Scheduling

Planning (cf. Section 2.1.2) and scheduling (cf. Section 2.1.1) are two rather related areas, and hence many actual problems involve both of them (cf. Section 2.1.3). However, these areas also present some differences. Both the similarities and the main differences are discussed in the current section.

2.1.1 Scheduling

The area of scheduling (Brucker et al., 2006; Pinedo, 2008) includes problems in which it is necessary to determine a schedule for a set of activities related by temporal and resource constraints. A schedule states (1) the start and end times of the activities to be executed and (2) the resource which is assigned to perform each activity. Since different activities may require the same resources, they may compete for limited resources (i.e., resource constraints). In general, the objective in scheduling consists of, given a set of activities, finding a feasible plan which satisfies both temporal and resource constraints. Resource constraints lead to establish a specific ordering between the activities which share the same resource, providing the problem with NP-hard complexity (Garey and Johnson, 1979). In scheduling problems several objective functions are usually considered

to be optimized, in most cases related to temporal measures, or considering the optimal use of resources.

In scheduling, an activity refers to a task which needs to be executed during a specific amount of time units, usually without interruption (i.e., preemptive scheduling), and using certain specific resources.

A quite general scheduling problem is called Resource-Constrained Project Scheduling Problem (RCPSP, cf. Brucker and Knust (2006)). RCPSPs are specified by a set of activities which are related by precedence constraints[1]. Moreover, for the execution of each activity, several units of many resources may be required. An extension of RCPSPs is the Multi-mode Resource-Constrained Project Scheduling Problem (cf. Drexl and Gruenewald (1993)). This problem is a RCPSP in which the activities can be executed in more than one operating mode, each one potentially using different resources, and usually presenting different values for certain properties, e.g., duration or cost of the activity.

In many scheduling problems, the activities are organized in jobs, i.e., sequences of activities which establishe precedence relations between the activities so that an activity can start only when all its predecessors have been executed.

Many variants of scheduling problems exist. Some of them are listed as follows (cf. Figure 2.1):

- Job Shop (cf. Brucker and Knust (2006); Pinedo (2008)): Each activity can only be executed using a specific resource.
- Flow Shop (cf. Brucker and Knust (2006)): It is a special case of the job shop problem in which each job is composed by exactly the same number of activities (which is equal to the number of resources). In this way, each job contains exactly one activity to be executed using each resource, and hence each job uses each resource exactly once. Moreover, all jobs use the resources in the same ordering.
- Flexible Job Shop (cf. Brandimarte (1993)): Many job centers exist, each one containing the same number of resources. In this way, an activity can be executed in any job center using the suitable resource.
- Cumulative Job Shop (cf. Nuijten and Aarts (1996b)): It is a generalization of the Job Shop in which the resources have a finite capacity and the activities may require several unities of several kinds of resources.
- Open Shop (cf. Pinedo (2008)): Unlike in job shop problems, in open shop problems the jobs do not have a predetermined fixed route.

[1] "Activity a precedes activity b" means that activity b cannot start before a is finished.

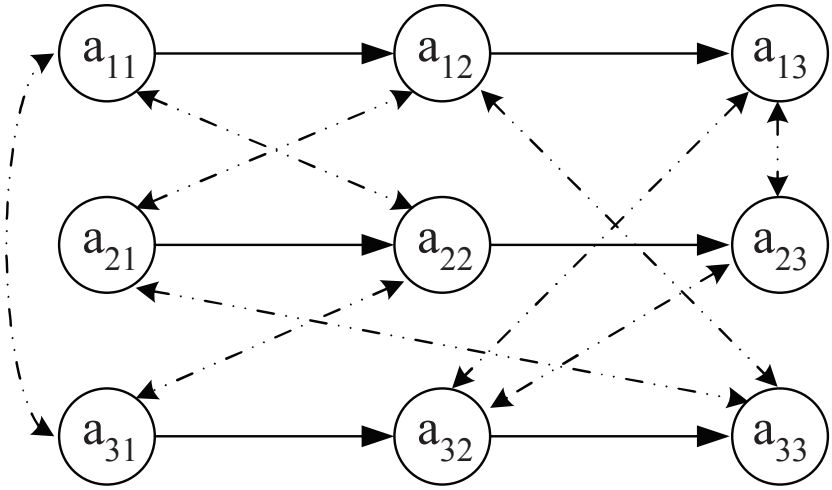

Figure 2.1: A representation of a job shop problem.

There are many typical objective functions to be considered in scheduling problems. Some of them are listed as follows:

- Makespan: It refers to the time in which the execution of all activities have finished.
- Tardiness: It refers to the delay of all jobs or activities regarding a specific due date.
- Total Weighted Tardiness: It consists of a generalization of the tardiness, in which $\sum_{j \in Jobs} w_j \times T_j$ is minimized, where w_j usually refers to an importance factor related to job j, e.g., holding cost per unit time, and T_j refers to the delay of job j regarding a specific due date.
- Number of Tardy Jobs: It refers to the number of jobs which do not meet their due dates.
- Total Weighted Completion Time: It consists on minimizing $\sum_{j \in Jobs} w_j \times C_j$, where w_j usually refers to an importance factor related to job j, and C_j refers to the completion time of job j.
- Objectives related to the use of the resources by the activities, e.g., balanced use of resources.

2.1.2 Planning

In a wider perspective, in Artificial Intelligence (AI) planning (Ghallab et al., 2004), the activities to be executed are not established a priori, hence it is necessary to select them from a set of alternatives and to establish an ordering. In most

cases, the specification of planning problems includes the initial state of the world, the goal (i.e., a predicate representing a set of possible final states) that must be reached, and a set of operators (i.e., actions) which can be applied to one state in order to reach another state. In general, the objective in planning consists of, given a set of available activities, generating a schedule by selecting and ordering a set of activities in a way that the resulting plan reaches a given goal. Furthermore, in planning problems, usually the optimization of certain objective functions is considered.

Taking the goals to be achieved into account, different planning strategies can be used for representing and reasoning about planning scenarios, e.g., Classical Planning (Fikes and Nilsson, 1971; Lekavy and Návrat, 2007), Hierarchical Task Network (Erol et al., 1994), Decision-Theoretic Planning (Joshi et al., 2011), Case-based Planning (Hammond, 1990) and Reactive Planning (Fernandes et al., 1983).

In order to reuse the same algorithms for solving different kinds of problems, and to solve the same problem using different algorithms, (1) domains for representing the problems, and (2) algorithms for solving the problems are specified in a separated way (domain-independent planning). For solving a specific problem, a domain-independent planner takes as input the problem specification and the domain information.

The first strategy which was proposed for representing and reasoning about planning scenarios was Classical Planning (Fikes and Nilsson, 1971; Lekavy and Návrat, 2007). The basic idea of Classical Planning consists of finding a sequence of actions which will modify the initial state of the world into a final state where the goal holds. The specification of Classical Planning problems is composed by:

- A set of literals from the propositional calculus which can be positive or negative and which represent the goal to be achieved.
- A set of literals from the propositional calculus which can be positive or negative and which represent the initial state, also known as initial conditions.
- A set of actions which are characterized by STRIPS operators. A STRIPS operator is a parameterized template used for stating a set of possible actions. Each action is composed by:
  - A set of preconditions: set of positive or negative literals which must be true for executing a specific action.
  - A set of effects: set of positive or negative literals which become true after the execution of the action.

As mentioned before, for executing an activity, all the literals included in the precondition of the activity need to be true. Therefore, at each stage, there is a set of possible activities to be executed which depends on the literals which are true in that moment (state of the world, i.e., a set of atoms or literals that define how the objects of the model relate to each other and their properties). Each time a specific activity is executed, the set of literals which are true changes, and hence the set of possible activities to be executed also changes. In this way, the state of the world evolves.

A solution for a planning problem is determined by a sequence of activities which reaches the final state from the initial state.

In such context temporal analysis is typically applied over the resulting schedules to figure out the temporal slack of the activities (Brucker and Knust, 2006; Gueorguiev et al., 2009), i.e., to calculate which activities can delay or advance their execution without affecting the completion time of the schedule. In this respect, different techniques such as CPM, PERT or Gantt charts (Gantt, 1913) can be used to perform this analysis in order to calculate the enactment plan (cf. Definition 1) related to a specific schedule. The same enactment plan can be related to different schedules, as shown in Figure 2.4.

Definition 1. *An **enactment plan** $P = (pid, Acts)$ is identified by pid and is composed of a set of activities $act \in Acts$ which are executed without preemption. Each activity act is a tuple <actid, Pred, dur, es, le, res> where: actid is an unique identifier in the enactment plan, Pred is the list of its precedence activities (i.e., those activities which must be executed before act), dur is the estimated duration of act, es is the earliest start time (i.e., the soonest that the activity act can start), le is the latest end time (i.e., the latest that the activity act can finish), and res is the resource which performs the activity act in the plan*[2].

Such definition is provided to formalize the concepts which already exist in the literature related to that term.

2.1.3 Integrating P&S

Planning and scheduling are rather related areas since both deal with the temporal planning of activities. The main difference between both areas is that in scheduling the activities to be planned are known and that it always involves the resource perspective, while in planning the activities which will be included in the plan need to be determined and resource constraints are not always considered.

[2]Note that, since activities are executed without preemption and the same resource cannot be used to perform more than one activity in parallel, there are implicit precedence relations between the activities which are executed by the same resource since the current approach does not allow a resource doing multiple activities in parallel.

Many works which combined P&S can be found, since several actual problems involve both of them. A problem involving P&S is characterized by: (1) there is a goal to be reached through the execution of a sequence of activities which are unknown a priori (planning), and (2) each of these activities has a specific duration and requires a specific resource to be executed, so that temporal constraints exist between the execution of activities, and certain (temporal) objective function needs to be optimized (scheduling).

Some of the extensions to scheduling that have been considered, such as alternative resources and process alternatives, lead to models that are closer to planning, as problems involving choice of actions are often regarded as planning problems (Smith et al., 2000).

Some planning techniques are not able to represent or reason with resources, metric quantities or continuous time. Moreover, planning techniques do not typically consider optimization. Therefore, there are many works which extend Classical Planning techniques in order to deal with resources (Drabble and Tate, 1994; Laborie and Ghallab, 1995), metric quantities (Koehler, 1998; Penberthy and Weld, 1994), and optimization criterions (Wolfman and Weld, 1999; Vossen et al., 1999). Furthermore, there exist works which extend planning techniques in order to allow working with continuous time and temporal constraints (Penberthy and Weld, 1994; Smith and Weld, 1999).

2.2 Constraint Programming

In such context, constraint programming (CP) (Rossi et al., 2006) (cf. Figure 2.2) supplies a suitable framework for modeling and solving problems involving P&S aspects (Salido, 2010). In order to solve a problem through CP, it needs to be modelled as a constraint satisfaction problem (CSP) (cf. Definition 2).

Definition 2. *A **CSP** $P = (V, D, C_{CSP})$ is composed of a set of variables V, a set of domains D which is composed of the domain of values for each variable $var_i \in V$, and a set of constraints C_{CSP} between variables, so that each constraint represents a relation between a subset of variables and specifies the allowed combinations of values for these variables.*

The given definition is provided to formalize the concepts which already exist in the literature related to CSP.

A solution to a CSP (cf. Definition 3) consists of assigning values to CSP variables.

Definition 3. *A **solution** $S =< (var_1, val_1), (var_2, val_2), ...(var_n, val_n) >$ for a CSP $P = (V, D, C_{CSP})$ is an assignment of a value $val_i \in dom_i$ to each variable $var_i \in V$.*

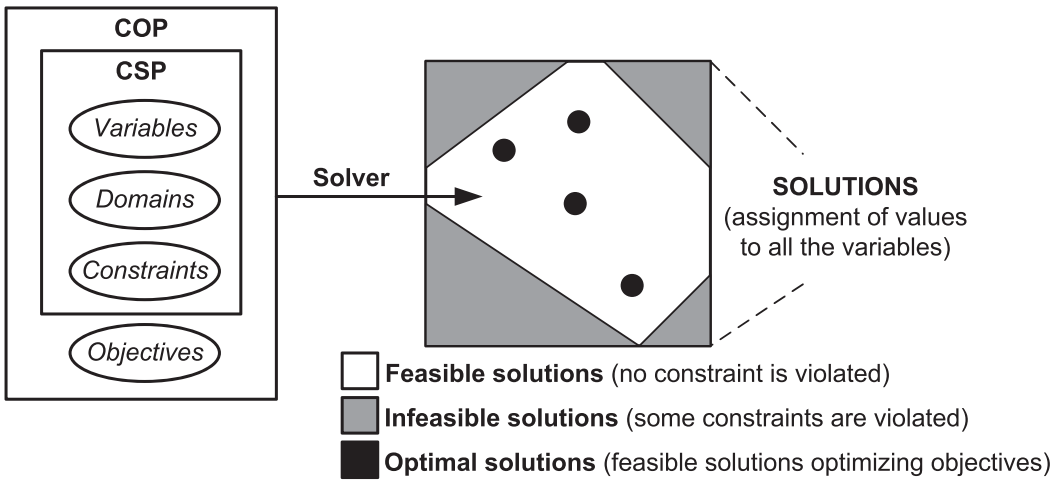

Figure 2.2: Schema of Constraint Programming.

A solution is **feasible** when the assignments variable-value satisfy all the constraints, i.e., a goal state is reached. In a similar way, a CSP is feasible if at least one feasible solution for this CSP exists. From now on, S^{var} refers to the value assigned to variable *var* in a solution S.

Similar to CSPs, constraint optimization problems (COPs, cf. Definition 4) require solutions that optimize an objective function.

Definition 4. *A COP* $P_o = (V, D, C_{CSP}, o)$ *related to a CSP* $P = (V, D, C_{CSP})$ *is a CSP which also includes an objective function o to be optimized.*

A feasible solution S for a COP is **optimal** when no other feasible solution exists with a better value for the objective function o.

Once a problem is modelled as a CSP, several goals can be pursued, e.g.:

- Finding any feasible solution for the CSP.
- Finding several feasible solutions for the CSP.
- Finding all the feasible solutions for the CSP.
- Finding the optimal (or optimized) solution for a COP.
- Finding a set of optimal (or optimized) solutions for a COP.

Example 1. *A classic problem which can be modelled as a CSP is the map coloring problem. This problem consists of coloring a map which is divided in a set of regions so that a color need to be assigned to each region, taking into account that regions sharing a boundary line do not have the same color and only specific*

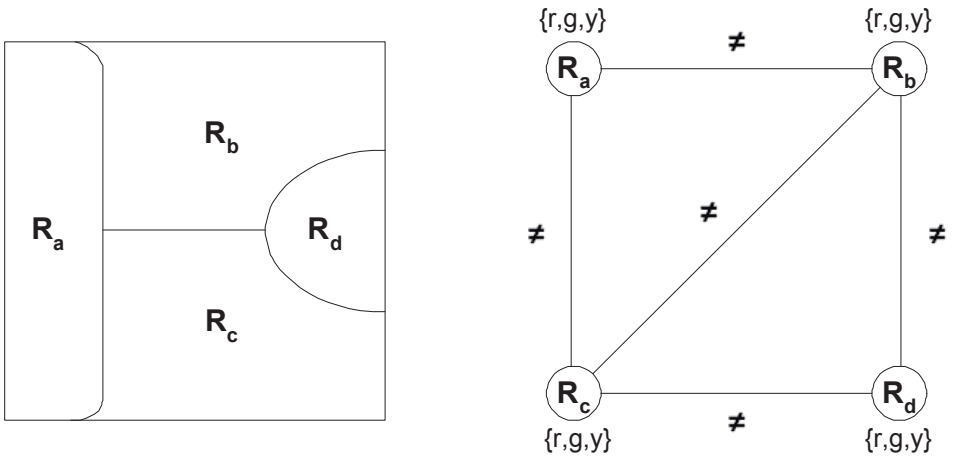

Figure 2.3: Map coloring problem.

colors can be used. The modelling of this problem as a CSP is made so that each region is a CSP variable, the domain of each variable is composed by the set of allowed colors, and the constraints establish inequality relations between the variables which represent adjoining regions. Figure 2.3 shows an example for this problem in which there are 4 regions, R_a, R_b, R_c and R_d, and 3 allowed colors, red (r), green (g) and yellow (y).

Constraint programming allows to separate the models from the algorithms, so that once a problem is modelled in a declarative way as a CSP, a generic or specialized constraint-based solver can be used to obtain the required solution. Furthermore, constraint based models can be extended in a natural way, maintaining the solving methods. Several mechanisms are available for solving CSPs and COPs (Rossi et al., 2006), which can be classified as search algorithms (i.e., for exploring the solution space to find a solution or to prove that none exists) or consistency algorithms (i.e., filtering rules for removing inconsistent values from the domain of the variables). In turn, search algorithms can be classified as:

- Complete search algorithms. Such algorithms (which are also called systematic algorithms) guarantee that a solution will be found if one exists, and can be used to show that a CSP does not have a solution and to find a optimal solution in COPs.

  The possible combinations of assignments of values to the CSP variables lead to a space state which can be represented by a tree o search graph. Each node of the search tree represents a partial assignment of values to a set of variables. The root node of the search tree represents the case in which any variable is instantiated, while the leaf nodes represent the cases in which all the variables are instantiated.

There are many systematic search algorithms, most of them are based on chronological backtracking (Mouhoub et al., 2003).

- Incomplete search algorithms. Incomplete search algorithms consist of exploring only certain regions of the state space so that, in general, the reach of a (optimal) solution can not be guaranteed. They are widely used due to the complete search usually requires a high cost. Local or stochastic search algorithms are examples of incomplete algorithms.

  Local search algorithms typically start generating a first solution of a given problem instance (in a random or a heuristic way), which may be infeasible, suboptimal or incomplete. This initial solution is iteratively improved so that the value for the objective function is optimized in the case of COPs, or the number of inconsistencies are reduced in the case of CSPs. In most cases, these algorithms finish after certain tries or iterations have been completed, or when the required solution is found.

  A wide scope of local search algorithms can be found in the literature, e.g., genetic algorithms (Mitchell, 1998), simulated annealing (Kirkpatrick et al., 1983), taboo search (Glover, 1989), and Greedy Randomized Adaptive Search Procedure (Feo and Resende, 1989, 1995). Different local search algorithms vary in the way in which improvements are achieved, and in particular, in the way in which situations are handled when no direct improvement is possible.

  Moreover, some algorithms combine systematic and local search techniques, e.g, Large Neighborhood Search (Pisinger and Ropke, 2010).

In this work P&S is applied to generate different possible enactment plans from the same constraint-based BP model through an incomplete search algorithm.

Since actual problems typically involve multiple conflicting objective functions, multi-objective constraint optimization problems (MO-COPs, cf Definition 5) are considered in the current work. The reader is referred to Ehrgott and Gandibleux (2003) for a review of the literature on MO-COPs.

Definition 5. *A MO-COP* $MP_o = (V, D, C_{CSP}, OFs)$ *related to a CSP* $P = (V, D, C_{CSP})$ *is a CSP which also includes a set of objective functions OFs to be optimized (maximized or minimized).*

Such definition is provided to formalize the concepts which already exist in the literature related to that MO-COP.

In multi-objective optimization problems, usually no unique optimal solution exists, but a set of Pareto optimal solutions (cf. Definition 6) can be found.

Definition 6. *Let $Sols$ be the set of all the solutions of a MO-COP MP_o which includes n objective functions, i.e, $OFs = OF_1, ..., OF_n$. Then, a solution $sol_1 \in Sols$ is **Pareto optimal** if $\nexists sol_2 \in Sols$ such that $\forall OF \in OFs : sol_2^{OF}$ is better or equal than sol_1^{OF} , i.e., for obtaining a feasible solution which improves one objective functions, at least another objective function needs to be deteriorated.*

Since many MO-COPs present NP complexity (Garey and Johnson, 1979), Pareto optimized solutions are considered (cf. Definition 7).

Definition 7. *Let $Sols$ be the set of all the solutions of a MO-COP MP_o and let $Sols_z \subseteq Sols$ be the subset of the solutions already explored at certain time. Then, a solution $sol_1 \in Sols_z$ is **Pareto optimized** if it is Pareto optimal regarding only the subset $Sols_z$, i.e., if $\nexists sol_2 \in Sols_z$ such that $\forall OF \in OFs : sol_2^{OF}$ is better or equal than sol_1^{OF}.*

To solve multi-objective optimization problems there are, basically, three approaches:

1. Defining a new objective function by combining the original objective functions, e.g., trough weighted-sum function (Leitmann, 1977; Zeleny, 1982; Chankong and Haimes, 1983). However, these approaches does not necessarily guarantee that the final solution will be neither acceptable (Koski, 1985; Stadler, 1995; Athan and Papalambros, 1996; Das and Dennis, 1997; Messac et al., 2000) nor Pareto optimized (Das and Dennis, 1997).

2. Working with stochastic algorithms like genetic, simulated annealing or ant colonies algorithms to obtain a set of Pareto optimized solutions. For example, previous works applied the simulated annealing technique (Smith et al., 2008; Suman, 2004) and evolutionary multi-objective optimization algorithms (Deb and Kalyanmoy, 2001; Shukla and Deb, 2007) for solving multi-objective optimization problems.

3. Optimizing one of the objective functions while constraining the other ones (e.g., ε-constraint method (Haimes et al., 1971)). These methods are based on optimizing only one of the objective functions while all the others objective functions are used to state additional constraints. Here, the main challenge is to select the proper bounds for the objectives which are not optimized. Each approach typically solves this issue with its own method. In general, each single-objective problem is solved several times by varying the value of the bound. The complete set of Pareto optimal solutions can be figured out if the bounds are adequately varied (Laumanns, 2006).

In this work, the ε-constraint method (Haimes et al., 1971) is applied since it appeared well suited for the purposes of this Thesis and typically provides good

results. In addition, in a previous approach (Jimenez-Ramirez et al., 2013a), such a technique was applied and it achieved good results. Furthermore, the base algorithm to be used in the ε-constraint method were developed and analyzed in previous approaches (Barba et al., 2013b,a) and results showed its effectiveness for improving performance. However, other promising multi-objective optimization techniques (e.g., stochastic algorithms) could be also applied.

2.2.1 Constraint Programming for Planning and Scheduling

Scheduling problems have been successfully addressed for a wide scope of applications using constraint-based techniques. Most of those problems can be modelled in a natural way, so that, since the actions are set, variables are chosen to correspond to the temporal unknowns (mainly start and end times) or to the ordering of tasks, and constraints gather precedence and resource constraints (Beck and Fox, 1998). Typical CSP modelings for the job shop problem states the start times of the activities as the variables of the CSP, and the constraints are divided in two groups:

- The precedence constraints are a set of inequalities involving the variables corresponding to the start times of the activities of the same job or related by precedence relations, and taking into account the durations of the activities.
- The resource constraints may be defined as disjunctions between the start time of the activities using the same resource. However, other approaches have been used, as representing the use of each resource by all the activities with global constraints, which may allow more efficient filtering algorithms.

Moreover, CP has been used in several recent AI planners (Nareyek et al., 2005; Vidal and Geffner, 2006; Tu et al., 2007; Gabriel and Grandcolas, 2009; Bao et al., 2011), since this paradigm is at the core for combining planning and scheduling techniques.

On the other hand, many constraint-based proposals for solving P&S problems exist in the literature, e.g., Timpe (2002); Liu and Jiang (2006); Gomes et al. (2006); Garrido et al. (2008); Moura et al. (2008); Garrido et al. (2009). Furthermore, several filtering algorithms for specialized scheduling constraints have been developed. Specifically, Beck and Fox (2000) and Bartak and Cepek (2010) model scheduling problems which include alternative and optional tasks respectively, together with their filtering rules. Moreover, the work Barták and Cepek (2008) proposes filtering rules for both precedence and dependency constraints in order to solve log-based reconciliation (P&S) problems in databases. In those studies, the precedence constraints signify the same as in P&S problems, while the dependency constraints are given between optional activities which can potentially

be included in the final schedule. The work Laborie et al. (2009) introduces new types of variables (time-interval, sequence, cumulative, and state-function variables) for modelling and reasoning with optional scheduling activities. In addition, Lombardi and Milano (2010) presents a set of filtering rules for cumulative constraint propagation when solving an extension of the resource-constrained project scheduling problem which includes time lags and uncertain, bounded activity durations. Furthermore, Monette et al. (2009) includes a constraint-based approach for the Just-In-Time Job Shop Scheduling, i.e., each activity has an earliness and a tardiness cost with respect to a due date. This approach includes a filtering algorithm which uses a machine relaxation to produce a lower bound, and dedicated heuristics. This work also includes pruning rules which update the variable bounds and detect precedence constraints.

2.3 Business Process Management

Nowadays, there exists a growing interest in aligning information systems in a process-oriented way (Weske, 2007) as well as in the effective and flexible management of business processes (cf. Definition 8) (Reichert and Weber, 2012). Organizations need to adapt to the new commercial conditions, as well as to respond to competitive pressures, considering the business environment and the evaluation of their information systems.

Definition 8. *A **Business Process** (BP) can be defined as a set of related structured activities whose execution produce a specific service or product required by a particular customer. These activities can be manual activities, other BPs, or even pieces of software.*

In order to use and manage business processes, business analysts need to specify the BPs through BP models (cf. Definition 9) by using a BP modelling language.

Definition 9. *A **Business Process model** consists of a set of activity models and execution constraints between them (Weske, 2007).*

In the literature, a wide spectrum of paradigms for BP modelling are presented, each one entailing different levels of accuracy in the BP elicitation, e.g., declarative and imperative paradigms. Such BP models are typically specified by hand using imperative languages like EPC or BPMN (BPMN, 2011). This way, a precise activity sequence which establishes how a given set of activities has to be performed is defined. These imperative models can be graphically represented using a BP graph (cf. Definition 10). The graph definition is introduced in Rosa et al. (2012).

Definition 10. *A **Graph** $G = (gid, N, Edges)$ is identified by gid and consists of a set of pairs of nodes $n \in N$, i.e., Edges. Each edge denotes a direct edge between two nodes in the graph. A node $n \in N$ is a tuple $< nid, l, t >$ where nid is an unique identifier of a node in the graph, l is its label, and t is its type.*

With relation to planning and scheduling, since there are several parallelism with BPM, an enactment plan (cf. Definition 1) can be represented as a graph. The activities of an enactment plan which are not preceded by any other activity are called initial activities. In a similar way, the activities which do not precede any other activity are called final activities. Therefore, regarding the precedence relations between the activities (stated by the *Pred* attribute, cf. Definition 1) and the parallelism that exists between activities which are executed by different resources (stated by the *res* attribute, cf. Definition 1) the schedule can be represented as a graph. In such graph, both a start node which precedes all of the initial activities and a final node which is preceded by all the final activities are additionally included (cf. Example 2).

Example 2. *Figure 2.4 (a) shows two schedules related to how to prepare a holiday where the activities* book a hotel, select the clothes *and* prepare the luggage *are considered. Since both schedules include the same activities, which are executed by the same resources and also in the same order, they result in the same enactment plan. As can be seen in the Gantt diagram related to the enactment plan (cf. Figure 2.4 (b)), the activity* Book hotel *presents 1 temporal unit of slack. In addition, Figure 2.4 (c) shows the related graph using BPMN*[3]*. This graph consists of the following 7 nodes (cf. Definition 10):* $<1, start, event>$, $<2, AND, gateway>$, $<3, book, activity>$, $<4, select, activity>$, $<5, AND, gateway>$, $<6, pack, activity>$ *and* $<7, end, event>$*; which are paired (cf. Definition 10) as follows:* $(1,2)$, $(2,3)$, $(2,4)$, $(3,5)$, $(4,5)$, $(5,6)$, *and* $(6,7)$.

Such definition of graph allows one to represent an imperative model (i.e., an enactment plan, cf. Definition 1) in many different languages, e.g., BPMN or EPC. As an example, the types of nodes (i.e., t) in BPMN language (BPMN, 2011) are 'activity', 'event', or 'gateway'. A node of type 'gateway' allows labels (i.e., l) 'AND', 'OR', 'XOR', etc., while 'event' nodes allow 'start' and 'end' labels (cf. Figure 2.5).

However, declarative BP models (e.g. constraint-based BP models, cf. Section 2.3.1) are increasingly used and their usage allows the user to specify what has to be done instead of having to specify how it has to be done. Some examples of BP models are shown in Figs. 2.7 and 2.8 which are explained later.

[3]For simplicity, role information is shown inside the activity boxes in the BP graph. In Rosa et al. (2012), a general solution for managing role information and other non-control-flow elements is shown.

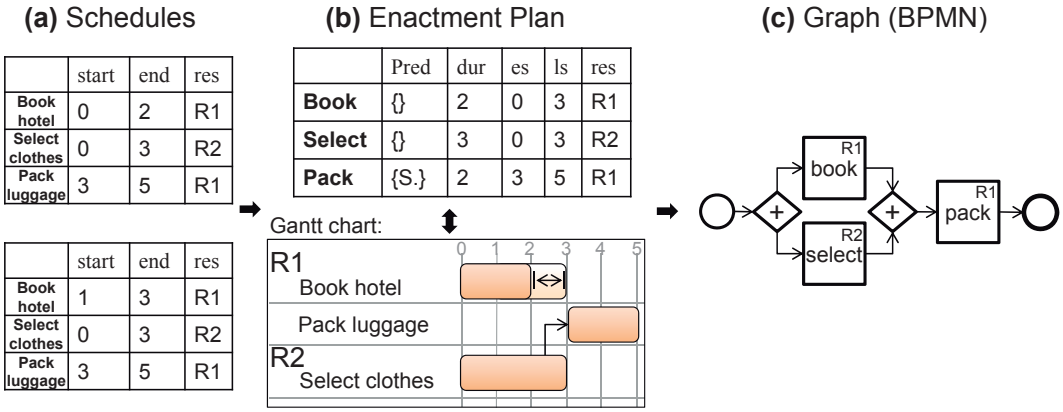

Figure 2.4: Relations between schedules, enactment plans, Gantt charts and graphs

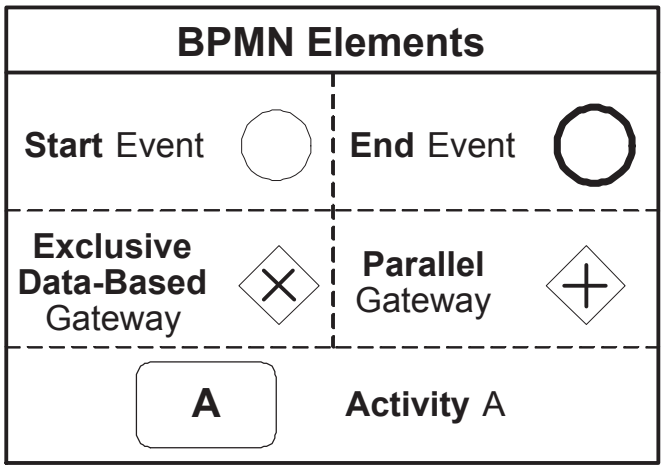

Figure 2.5: Basic elements of BPMN.

The modelling of the processes plays an important role in the overall management of BPs (Business Process Management, BPM, cf. Definition 11) (Davenport, 1993; Georgakopoulos et al., 1995). In the current business world, the economic success of an enterprize increasingly depends on its effectiveness in the management of its BPs, and hence BPM is an interesting research area which is being widely analyzed nowadays. In a related way, BPM Systems (cf. Definition 12) are software tools that support the management of the BPs.

Definition 11. ***Business Process Management*** *(BPM) can be seen as supporting BPs using methods, techniques, and software to design, enact, control and analyze operational processes involving humans, organizations, applications, documents and other sources of information (van der Aalst et al., 2003).*

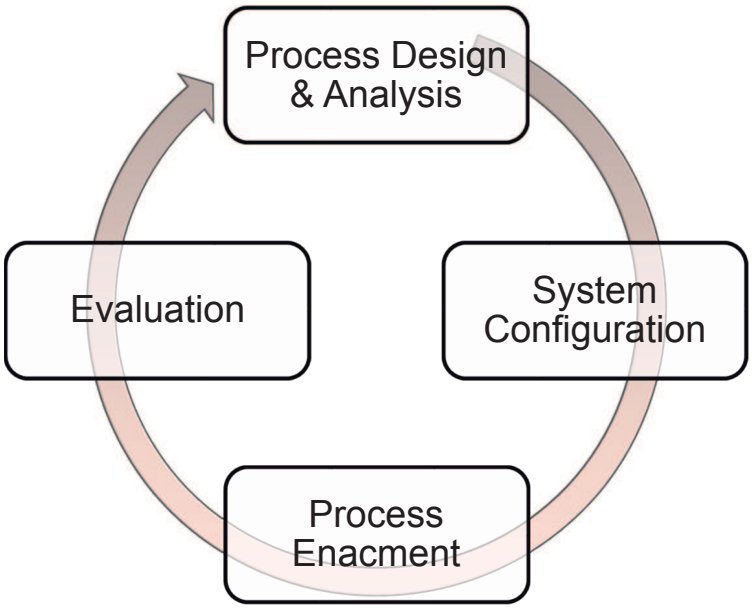

Figure 2.6: Typical BPM Life Cycle.

Definition 12. *A **Business Process Management System** (BPMS) is a generic software system that is driven by explicit process representations to coordinate the enactment of business processes (Weske, 2007).*

Such management generally follows a strict methodology to ensure the quality of the information systems which are created. Traditional BPM Life Cycle (Weske, 2007) (cf. Figure 2.6) includes four phases which are related to each other. These stages are organized in a cyclical structure which shows the logical dependencies between them:

- Process Design & Analysis: In this phase, BPs are identified, reviewed, validated, and represented by BP models, so that the informal BP description is formalized using a particular BP modelling notation. Two steps are considered to create a BP model: (1) draw an initial BP model, and (2) improve this initial model by simulation or BP redesign techniques. Traditionally, this phase is mostly a human activity. In some cases, process models can be verified against inconsistencies and errors (van Dongen, 2007).

- System Configuration: In this phase, BP models are implemented by configuring a BPM system. There are different ways to do so, e.g., by stating a set of policies and procedures. Service-oriented architectures as well as web services for their implementation have gained increasing popularity for BPMSs implementations recently. Moreover, data-driven approaches to the

flexible enactment of BPs are considered for enactment of human interaction BPs using data dependencies to control process enactment.

- Process Enactment: After completing system configuration phase, BP instances can be enacted. In this phase, the BPMS controls the execution of BP instances as defined in the BP model. As execution proceeds, the enactment information must be analyzed due to the possible appearance of unexpected events.

- Evaluation: In this phase, information regarding the BP enactment is evaluated in order to identify and improve the quality of the BP model and their implementations. Traditionally, enactment logs are analyzed by using BP activity monitoring and BP mining techniques.

After the Evaluation phase, BP models are corrected and improved in the BP Design & Analysis phase if necessary by considering the evaluation information, and hence closing the cycle which shows the logical dependencies between the phases of the BPM life.

In this Thesis Dissertation, the BP Design & Analysis phase is widely analyzed since this phase plays an important role in the BPM life cycle for any improvement initiative, and it greatly influences the remaining phases of this cycle. Specifically, constraint-based BP models (cf. Section 2.3.1) and configurable BP models (cf. Section 2.3.2) are analysed.

2.3.1 Constraint-based BP Models

Recently, constraint-based approaches have received increased interest (Vanderfeesten et al., 2008; Pesic, 2008; Westergaard and Maggi, 2012; Montali et al., 2013) since they suggest a fundamentally different way of describing BPs which seems to be promising in respect to the support of highly dynamic processes (Vanderfeesten et al., 2008; Pesic, 2008). Irrespective of the chosen approach, requirements imposed by the BPs need to be reflected by the process model. This means that desired behavior must be supported by the process model, while forbidden behavior must be prohibited (Pesic et al., 2007; van der Aalst et al., 2009; Montali, 2009). While executable process models specify exactly how things have to be done[4], declarative process models focus on what should be done. In the current approach, declarative BP specifications are considered since, as stated, the specification of process properties in a declarative way is an important step towards the

[4]With the term executable models it is referred to imperative models which are rather strict and which represent only one enactment plan (or at most only few decision points are included).

flexible management of BPMs (van der Aalst et al., 2009). In the literature, several rule-based and constraint-based languages for declarative BP modeling are proposed (e.g., van der Aalst and Pesic (2006a); Dourish et al. (1996); Wainer and De Lima Bezerra (2003); Lu et al. (2006)). This Thesis Dissertation uses the language Declare (also known as ConDec) (Pesic, 2008; Pesic et al., 2007) for the BP control-flow specification. Declare is considered to be a suitable language, since it allows the specification of BP activities together with the constraints which must be satisfied for correct BP enactment and for the objective to be achieved (cf. Definition 13).

Definition 13. *The **goal** of a BP is specified through the constraints which must be satisfied during the BP enactment.*

Moreover, Declare allows to specify a wide set of BP models of varied nature, flexibility and complexity in a simple way. In addition, Declare has been widely referenced in the past years in the context of BPs (Ly et al., 2008; Montali, 2009; Lamma et al., 2007; Chesani et al., 2009). Declare is based on constraint-based BP models (cf. Definition 14), i.e., including information about (1) activities that can be performed as well as (2) constraints prohibiting undesired process behavior.

Definition 14. *A **constraint-based BP model** $CM = (A, C_{BP})$ consists of a set of activities A, and a set of constraints C_{BP} prohibiting undesired execution behavior. Each activity $a \in A$ can be executed arbitrarily often if not restricted by any constraints.*

Such definition is provided to formalize the concepts which already exist in the literature related to constraint-based BP model.

Constraints can be added to a Declare model to specify forbidden behavior, restricting the desired behavior. For this, Declare proposes an open set of templates which can be divided into 4 groups:

1. **Existence** templates: unary relationships concerning the number of times one activity is executed, e.g., Exactly(N,A) specifies that A must be executed exactly N times.

2. **Relation** templates: positive binary relations used to impose the presence of a certain activity when some other activity is performed, e.g., Precedence(A,B) specifies that to execute activity B, activity A needs to be executed before.

3. **Negation** templates: negative relationships used to forbid the execution of activities in specific situations, e.g., NotCoexistence(A,B) specifies that if B is executed, then A cannot be executed, and vice versa.

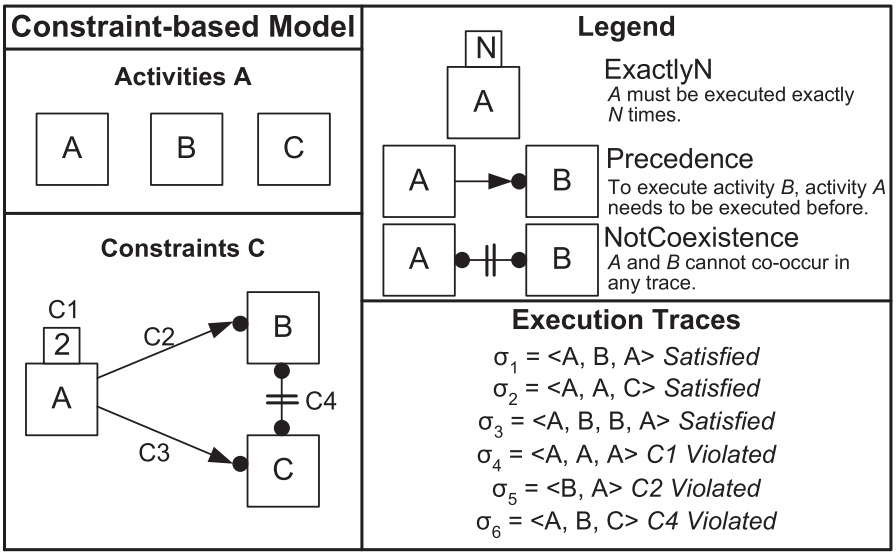

Figure 2.7: Simple constraint-based model for a set of activities.

4. **Choice** templates: n-ary relationships expressing the need of executing activities belonging to a set of possible choices, e.g., ExactlyChoice(N,{A,B,C}) specifies that exactly N activities of the set {A,B,C} must be executed.

In Declare, while unary relationships describe constraints related to one activity (e.g., existence constraints), binary constraints describe relationships between activities (e.g., precedence constraints). Binary templates are composed by a source activity (cf. Definition 15) and a sink activity (cf. Definition 16), which correspond to the beginning and the end of the arrow related to the specific template in the graphical notation of Declare, respectively.

Definition 15. *A source activity of a binary template is an activity which appears in the first parameter of the template. For templates which state precedence relations between activities, a source activity is a predecessor activity.*

Definition 16. *A sink activity of a binary template is an activity which appears in the second parameter of the template. For templates which state precedence relations between activities, a sink activity is a successor activity.*

Figure 2.7 shows a simple constraint-based model which is composed by activities *A*, *B*, and *C*, and constraints *C*1 (ExactlyN(A)), *C*2 (Precedence(A,B)), *C*3 (Precedence(A,C)), and *C*4 (NotCoexistence(B,C)).

In Declare, binary constraints can be extended by defining branched templates, as described in Pesic (2008). The branched templates for the binary templates can be established between several BP activities in the following way:

- The branched constraint is established between several source activities and one sink activity, so that the relation is given between *at least* one of the sources and the sink[5].

- The branched constraint is established between one source activity and several sink activities, so that the relation is given between the source and *at least* one of the sinks[6].

In Declare (van der Aalst and Pesic, 2006a), the fact of considering atomic activities is recognized as being a major problem. Similar to Declare, the languages ConDec++ (Montali, 2009) (an extension of Declare) and Saturn (Demeyer et al., 2010) are constraint-based workflow definition languages based on LTL which, unlike Declare, consider non-atomic activities that can be started, completed or cancelled at a later time, and overlapped with other activities.

Once a BP is modelled through a constraint-based modelling language, the BP can be executed. As the execution of a constraint-based model proceeds, information regarding the executed activities is recorded in an execution trace (cf. Definition 17).

Definition 17. *Let $S = (A, C_{BP})$ be a constraint-based process model (cf. Definition 14). Then, a **trace** $T =< e_1, e_2, ...e_n >$ is composed of a sequence of starting and completing events regarding activity executions a_i, $a \in A$, i.e., events can be:*

1. *$start(a_i, t, r)$, i.e., the i-th execution of activity a is started at time t by the resource $r \in Res$.*

2. *$comp(a_i, t)$, i.e., the i-th execution of activity a is completed at time t.*

A process instance (cf. Definition 18) represents a concrete execution of a constraint-based model and its execution state is reflected by the execution trace.

Definition 18. *Let $S = (A, C)$ be a constraint-based process model with activity set A and constraint set C. Then: A **process instance** $I = (S, \sigma)$ on S is defined by S and a corresponding trace σ.*

A running process instance I is in state **satisfied** if its current partial trace σ satisfies all constraints stated in C. Furthermore, an instance is in state **violated**,

[5]These branched templates consider only the disjunction of conditions related to the sources, since the conjunction can be obtained by including the associated non-branched template between each source and the sink activity.

[6]These branched templates consider only the disjunction of conditions related to the sinks, since the conjunction can be obtained by including the associated non-branched template between the source and each sink activity.

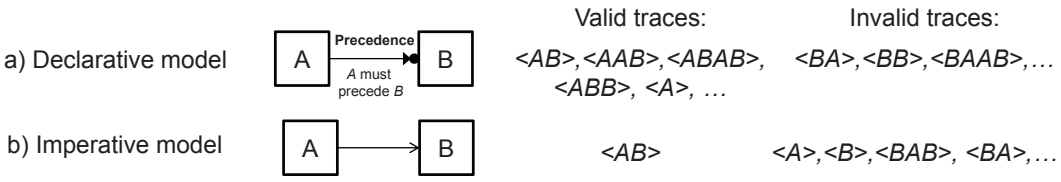

Figure 2.8: Increased flexibility of declarative models versus executable models

if the partial trace violates all constraints stated in C and there is no suffix that can be added to satisfy them (cf. Example 3).

Considering a constraint-based model and a specific related process instance, only certain activities are enabled to be executed next (cf. Definition 19, Example 3).

Definition 19. *Let $S = (A, C)$ be a constraint-based process model with activity set A and constraint set C, and $I = (S, \sigma)$ be a corresponding process instance with partial trace σ. Then: An activity a_i of instance I is **enabled** at time T iff a_i can be started and the instance state of I is not violated afterwards; i.e., for $\sigma =< e_1, e_2, ...e_n >$, it is obtained $\sigma'_l =< e_1, e_2, ...e_n, start(a_i, Rj_k, T) >$ afterwards and instance (S, σ') is not in state violated.*

Example 3. *Figure 2.7 includes examples of traces of satisfied and violated instances[7] for a constraint-based model. For the partial trace σ_1 of Figure 2.7, B is enabled, while A is not enabled due to C1, and C is not enabled due to C4.*

Due to their flexible nature, there are different ways to execute a constraint-based BP model in such a way that all constraints are fulfilled, i.e., the process goal is reached (cf. Definition 13 and Example 4).

Example 4. *Figure 2.8(a) shows a constraint-based BP model where traces[8] <AAB>, <AB>, <ABAB>, <ABB>, <A> are some of the valid ways of executing such model, while traces <BA>, <BB>, <BAAB> are invalid since A must precede B. In contrast, Figure 2.8(b) shows an executable model where there is only one valid execution trace, <AB>.*

The different valid execution alternatives related to a specific constraint-based BP model, however, can greatly vary in respect to their quality, i.e., how well different performance objective functions (cf. Definition 20) can be achieved.

[7] For the sake of clarity, only completed events for activity executions are included in the trace representation.

[8] For the sake of clarity, traces represent sequences of activities, i.e., no parallelism is considered in the examples. Moreover, only completed events for activity executions are included in the trace representation.

Definition 20. *The **objective function** of a BP is the function to be optimized during the BP enactment, e.g., minimization of overall completion time.*

Many real scenarios require the optimization of multiple objective functions. Thus, the automatic generation of a multi-objective optimized configurable BP model from a constraint-based BP model by applying constraint programming for Planning and Scheduling (P&S) the BP activities is suggested.

2.3.2 Configurable Business Process Models

Typically, different BP models (cf. Definition 9), also called *variants*, can be performed in scenarios which entail high variability, e.g., human resource management, clinical guidelines and financial accounting. Some enterprise system vendors manage that variability through general reference models, also call *configurable* BP models, which cover all the different variants. Generally, configurable BP models allow analysts to understand what these variations share, what their differences are, and why and how these differences occur (Rosemann and van der Aalst, 2007). Before such a general model can be used, it requires to be concretized to the individual context of the target organization. For this, a new phase, namely *configuration & individualization*, is defined in the BPM life cycle after the process design & analysis phase (cf. Figure 2.6) (La Rosa et al., 2008). At configuration-time a domain expert should select the most appropriate BP model depending on the context (e.g., business regulations, objective functions, etc.) (Gottschalk et al., 2008). Then, the selected BP model can be enacted and the remaining parts of the configurable BP model (i.e., those parts which have not been selected for execution) are not executed.

Basically, there are two ways for creating configurable BP models:

- By including possible adaptations (e.g., *hiding* and *blocking* methodology) (Gottschalk and Jansen-vullers, 2006; der Aalst et al., 2006; Gottschalk et al., 2008). It can be done from scratch or from an existing BP model. Some elements of the configurable BP model (e.g., activities and data) can be set to optional and, therefore, highlighting which configurations are possible or not (Gottschalk et al., 2008).

- By merging some BP models related to the same or similar goals which already exist (Rosa et al., 2012). In that case, the source BP models need to be compared and merged, which might result in a tedious, time-consuming and error-prone process if it is performed by hand (Rosa et al., 2012). To overcome these problems, there exist approaches focused on automatically merging different BP models in a configurable BP model (Rosa et al., 2010, 2012).

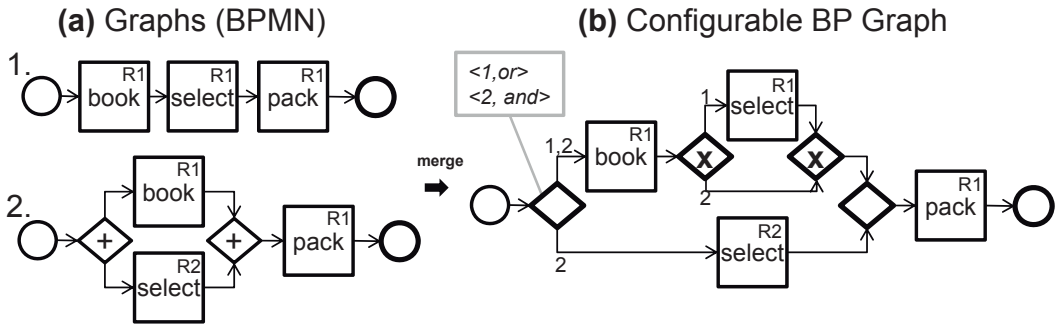

Figure 2.9: Two enactment plans (a) are merged into a single configurable BP model (b)

Configurable BP models can be represented by configurable BP graphs, which are defined (cf. Definition 21) based on Rosa et al. (2012).

Definition 21. *A **Configurable BP Graph** $CG = (G, E2I, N2LI)$ consists of: (1) a graph, $G = (gid, N, Edges)$ (cf. Definition 10), (2) a function $E2I$ that maps each edge $e \in Pairs$ to a set of process graph identifiers (i.e., $E2I$ identifies which branches of CG belong to each source graph which is merged in CG), (3) a function, $N2LI$ that maps each node $n \in N$ to a set of pairs $< gpid, l >$ where $gpid$ is a graph identifier and l is the label of node n in graph $gpid$ (i.e., $N2LI$ identifies which nodes, with the corresponding label, belong to each graph which is merged in CG).*

In the this Thesis, configurable BP models are created by using the process merger tool presented in Rosa et al. (2012) after being adapted to work with BPMN. This tool is based on a merging algorithm which analyzes the similarities of the input models (i.e., the graphs) and creates a new model (i.e., the configurable BP graph) which includes configuration nodes for those points where the input models are different. Therefore, each branch and node of the configurable BP model can be related either to one or more graphs. To store these relations, each branch/node of the configurable BP graph includes identifiers related to the corresponding plan (i.e., $E2I$ function). In addition, nodes also store the associated label related to each identifier (i.e., $N2LI$ function). Since a configurable BP model includes different graphs, it is considered that a configurable BP model includes different BP models (cf. Example 5).

Example 5. *Figure 2.9 shows 2 graphs which are merged into a configurable BP model* [9]*. The first gateway in Figure 2.9(b) is a configurable node which corresponds to an 'OR' gateway in the process 1 and an 'AND' gateway in the process 2.*

[9] As there is not ambiguity, some labels are not shown (i.e., they are the same as in the branch).

Questionnaires

Questionnaire models (Rosa et al., 2009) are typically created by the analysts to support the user during the configuration (i.e., individualization) of the configurable BP models. The main benefits of using them are: (1) they guide the user in such a way that choices are presented in a proper order and (2) they avoid invalid configurations which may lead to errors.

Typically, questionnaires are manually created by an analyst whereby each question is related to boolean *facts* which are associated to configuration *actions* (Rosa et al., 2009). Each time a question is answered, an action which configures a part of the configurable BP model is fired. The sequence of answers given to the different questions will individualize the configurable BP model in such a way that a single variant is selected before run-time to be executed.

Unlike previous approaches which deal with questionnaires, this Thesis:

- Automatically creates the questionnaires (i.e., defining facts and actions are not longer needed).
- The questionnaires which are created are intended to individualize the configurable BP models during run-time (cf. Chapter 4).

2.4 Dealing with the Uncertainty

When modeling and solving business problems, usually it is assumed that there is a complete and exact description of the problem, and that there is no change between the initial description of the scenario and the real scenario in which the solution is applied. These two assumptions do not hold for many practical applications since uncertainty is typically present in most real scenarios (e.g., the beauty salon scenario described Section 5.2).

The sources of uncertainty can be quite different, e.g., related to the imprecise knowledge of the system or due to external events (cf. Example 6).

Example 6. *For example, the full set of jobs to be scheduled in a factory can be unknown in advance; the durations of activities may vary from the initial estimations; there may be resource breakdowns; the availability of workers, machines or raw material, may be not guaranteed; or the weather conditions may affect to the validity of the initial plan.*

Therefore, mechanisms to deal with uncertainty are required for real systems. The main requirements that have been proposed when handling with uncertainties and changes are Verfaillie and Jussien (2005):

1. To limit as much as possible the need for successive online problem solving.

2. To limit as much as possible changes in the produced solutions when the first approach fails.

3. To limit as much as possible the computing time and resources that are necessary for online problem solving when the first approach fails.

4. To keep producing consistent and optimal solutions.

Different formalisms based in CSP (cf. Definition 2) have been proposed to represent the uncertainty in their models. Dechter and Dechter (1988) introduced dynamic constraint satisfaction problems (DCSP) as a sequence of CSP, each one resulting from small changes from the previous one. Other extensions from the CSP framework are the conditional constraint satisfaction problems (CCSP) (Sabin and Freuder, 1998), the open constraint satisfaction problems (OCSP) (Faltings and Macho-Gonzalez, 2005), the mixed constraint satisfaction problems (Fargier et al., 1996), the probabilistic constraint satisfaction problems (PCSP) (Fargier and Lang, 1993), the stochastic constraint satisfaction problems (SCSP) (Walsh, 2002), fuzzy constraint satisfaction problems (Dubois and Prade, 1993), or the branching constraint satisfaction problems (BCSP) (Fowler and Brown, 2000). In addition, L. Climent and Barber (2014) an algorithm for solving CSPs subject to uncertainty which looks for both stable and robust solutions.

According to the different frameworks that deal with uncertainty, a variety of solving methods for obtaining solutions have been proposed. In the surveys by Verfaillie and Jussien (2005), Brown and Miguel (2006) and Herroelen and Leus (2005), a wide number of proposals are collected, from both reactive and proactive approaches.

In literature (Verfaillie and Jussien, 2005), two classes of methods have been proposed to deal with uncertainties and changes:

- Reactive methods, which aim at reusing solutions or reasoning. They are applied when solutions are not longer valid. They may find new solutions or repair the previous ones that have been invalidated.

- Proactive methods, which aim at producing robust or flexible solutions. They may use knowledge about the uncertainties and changes in order to produce solutions that will resist as much as possible these changes.

In the context of proactive methods, flexibility and robustness concerns have received increasing attention in last years (Aissi and Roy, 2010; Stevenson and Spring, 2007; de Haan et al., 2011; Golden and Powell, 2000; Gueorguiev et al.,

2009; Cicerone et al., 2012; L. Climent and Barber, 2014), also in the context of BPs (Reichert and Weber, 2012; Schonenberg et al., 2008a).

Although flexibility and robustness are typically used to evaluate techniques or tools which cope with the natural uncertainty of real scenarios, literature related to adaptability-like topics acknowledges that there is not a simple and general definition for these terms. Some approaches even define different terms with similar definitions (Aissi and Roy, 2010; Golden and Powell, 2000), which leads to misunderstandings. As shown in de Haan et al. (2011), flexibility and robustness frequently appear on research articles since 1991. Between dozens of existing definitions of flexibility and robustness, in this work a representative set of them is selected in order to highlight their commonalities and differences.

On the one hand, flexibility was defined in the 80's as "the ability of an organization to adapt to [...] changes that [...] impact on the organizations performance" (Aaker and Mascarenhas, 1994). It was supported by (Upton, 1994) in the 90's which defines flexibility as "an organisation's ability to change [...] with little penalty [...]". A more recent definition (Schonenberg et al., 2008a), which applies this term to BPs, defines it as "the ability to deal with [...] changes, by varying or adapting those parts of the business process that are affected by them, whilst retaining the essential format [...]". In accordance with Reichert and Weber (2012); Schonenberg et al. (2008a); Golden and Powell (2000); Upton (1994); Aaker and Mascarenhas (1994), the term flexibility is defined as follows (cf. Definition 22):

Definition 22. ***Flexibility*** *is the capability to adapt a plan to external events in order to achieve a goal (i.e., to change the original plan to a new plan which generally has a different performance but achieves the same goal).*

Note the active feature of the verb *adapt* - the flexibility is an active ability (cf. Example 7).

Example 7. *A person who is going to the cinema wearing summer clothes when it is sunny but the forecast is uncertain increases the flexibility taking a foldable raincoat in a handbag. This way, taking a raincoat makes adaptation to the weather possible, and hence, the flexibility is increased (note that changing the clothes is necessary only if it rains).*

On the other hand, robustness was defined in the 70's as "the ability to respond successfully to unforeseen environmental changes" (Eppink, 1978) [10], which was recently supported by de Haan et al. (2011). In P&S, Jensen (2001) states that "robustness means that the schedule is still acceptable if small delays happen during schedule execution". In addition, similar definitions can be found in complex

[10] Actually, instead of robustness, the work Eppink (1978) uses the terms "passive" or "internal" flexibility.

systems which defines robustness as ”the maintenance of some desired system characteristics despite fluctuations in the behavior of its component parts or its environment” (Carlson and Doyle, 2002). In the area of project management, Gueorguiev et al. (2009) considers that ”a highly robust project is one with a lot of built-in flexibility; ’slack’ capacity that can be taken up in an emergency”. Therefore, the term robustness is defined (Jensen, 2001, 2003; Gueorguiev et al., 2009; Cicerone et al., 2012; Eppink, 1978; de Haan et al., 2011; Carlson and Doyle, 2002) as follows (cf. Definition 23):

Definition 23. ***Robustness*** *is the capability of a process to withstand external events in order to prevent undesirable impacts (i.e., changing the plan is not required, and hence, the same performance after the occurrence of the events is reached).*

In contrast to flexibility, robustness is considered a passive ability (cf. Example 8).

Example 8. *The same person of Example 7 increases the robustness wearing mid-season clothes instead of summer clothes since the mid-season clothes can withstand good and bad weather, and hence, the robustness is increased (note that, unlike in Example 7, in this case changing the clothes is not required).*

Typically, increasing the robustness decreases the performance, e.g., although the mid-season clothes can withstand both sunny and rainy weather, summer clothes and raincoats perform better under sunny and rainy conditions respectively. However, flexibility and robustness are not opposed (Jensen, 2001) but can be increased in a coordinated way, i.e., the flexibility can be increased by considering different alternatives while the robustness can be increased by providing such alternatives with the capacity of withstanding a higher uncertainty.

As shown above, definitions given in literature for both terms differ depending on the researchers in such a way that no formal definitions have been standardized. For this, this Thesis proposes quantitative definitions for both flexibility and robustness to measure how a system deals with the uncertainty of real scenarios (cf. Chapter 4).

Chapter 3

From Constraint-based BP Models to Multi-objective Optimized BP Enactment Plans

3.1 SDeclare 1.0: Extending Declare by Including Resource Reasoning, Temporal and Data Constraints

To specify the processes in a declarative way, Declare (Pesic, 2008) is used as basis (cf. Section 2.3.1). Motivated by requirements described in literature (Montali, 2009; Westergaard and Maggi, 2012) as well as the necessities of the case studies we have conducted (cf. Chapter 5), in this chapter a first extension of Declare is defined, resulting in the first version of the SDeclare language (such language is further extended in Chapter 4). Besides extending Declare with resource reasoning and estimates for activity durations (which are partially covered in Barba and Del Valle (2011)), SDeclare supports activities with an open set of attributes and alternative resources (cf. Definition 24), and choice, temporal and data constraints.

Definition 24. *A **S-Activity** SAct = (a, Res, Atts) represents a S-Activity a (cf. Definition 14) which can be performed by any resource included in Res*[1]*, and which has a set of attributes associated Atts (e.g., duration and profit). The set Atts is composed of tuples* <att, value>.

[1]This allows activities to be performed by alternative resources, whereas in previous works (cf. (Barba and Del Valle, 2011)) only one resource can be assigned to each activity.

One important aspect when modeling using SDeclare is that a S-Activity in SDeclare represents multiple executions of it. Therefore multiple process instances are allowed if not restricted by constraints over their activities. In a SDeclare process model (cf. Definition 25), all the previously stated extensions are considered (cf. Example 9).

Definition 25. *A **SDeclare process model** SDM = (SActs, Data, C_{BP}, AvRes, OFs) related to a constraint-based process model CM = (A,C_{BP}) (cf. Definition 14) is composed of (1) a set of S-Activities (cf. Definition 24) $SActs = (a, Res, Atts)$ related to each $a \in A$, (2) problem data information Data (which is the information which influences the process execution), (3) a set of SDeclare constraints C_{BP} (which relates activities included in Acts and/or the data included in Data), (4) a set of available resources AvRes (which is composed of tuples <role,#role> which includes for each role the number #role of available resources[2]), and (5) a set of the objective functions OFs to be optimized (cf. Definition 20).*

Example 9. *Figure 2(A) shows a simple SDeclare model[3] (cf. Definition 25) where:* $SActs = \{(A, <R1>, <<att_1, 2>, <att_2, 6>>), (B, <R2>, <<att_1, 2>, <att_2, 2>>), (C, <R1, R2>, <<att_1, 2>, <att_2, 3>>), (D, <R1, R2>, <<att_1, 3>, <att_2, 2>>)\}$; $Data = \{\}$; $C_{BP} = \{ exactly(1, A), exactly(2, B), succession(A, B), response(A, B), negate-response(B, C), precedence(C, D) \}$; $Res = \{(R1, 2), (R2, 2)\}$; *and* $OFs = \{OF_1, OF_2\}$.

The basic SDeclare templates, extending the Declare templates (van der Aalst and Pesic, 2006a), together with its formal specification through constraints and some examples of valid and invalid traces are listed in **Appendix A**.

3.1.1 Resource Reasoning

To support the direct reasoning with resources (which is not possible in Declare) Declare is extendeded by including: (1) alternative resources for executing each S-Activity (cf. *Res* in Definition 24), and (2) the set of available resources (cf. *AvRes* in Definition 14). In this way, SDeclare directly supports the most common workflow resource pattern, i.e., the role-based distribution (Russell et al., 2005), which also supports the cases study. This pattern models the ability to specify at design-time one or more roles which will be assigned to the instances of an activity at run-time. Note that, besides the role-based distribution pattern, SDeclare is open to support further resource patterns (Russell et al., 2005) by including the related constraints in the proposed CSP model (cf. Definition 14). However, as

[2]The role-based allocation pattern (Russell et al., 2005) is considered.

[3]We extend Declare tool (Declare, 2011) (i.e, a workflow management system that can be used to specify Declare models) to allow specifying SDeclare models.

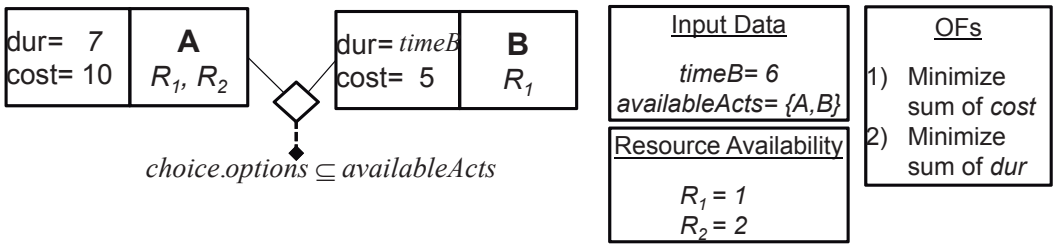

Figure 3.1: Example of SDeclare process model.

mentioned, this work is focus on the role-based distribution pattern, which is the one required for modelling the considered case study.

The information related to resource availabilities can be unknown until starting the BP enactment. Since this information is independent of the BP-Activities, it can be changed without affecting the specification of the activities, and vice versa. This is not a problem for the current proposal since static information (i.e., the control-flow and resource constraints) is complemented with more changing information (i.e., the estimates), and finally the most dynamic information (i.e., information about resource availabilities) is included. In this way, with the current approach, the configurable BP model can be automatically generated just before starting the BP enactment by considering the actual values of the resource availabilities and estimates.

3.1.2 Temporal and Data Constraints

To support increased expressiveness of Declare templates, it is extended by considering temporal and data constraints. In this way SDeclare allows to specify temporal constraints in a similar way as (Montali, 2009; Westergaard and Maggi, 2012), i.e., all the Declare constraints have been extended to support temporal modifiers, e.g., the SDeclare constraint *Precedence(A, B, [5, 10])* states that for starting the execution of activity B, activity A needs to be finished between 5 and 10 time units before. Furthermore, Declare is extended by including data constraints in a similar way as (Montali, 2009).

Therefore, input data can be related to (1) activity attributes, e.g., in Figure 3.1, the duration of the activity *B* is specified by the input data, (2) resource availability, i.e., the number of available resources of a role, and (3) SDeclare constraints, e.g., in Figure 3.1, the selection of the choice constraint depends on the input data *availableActs*.

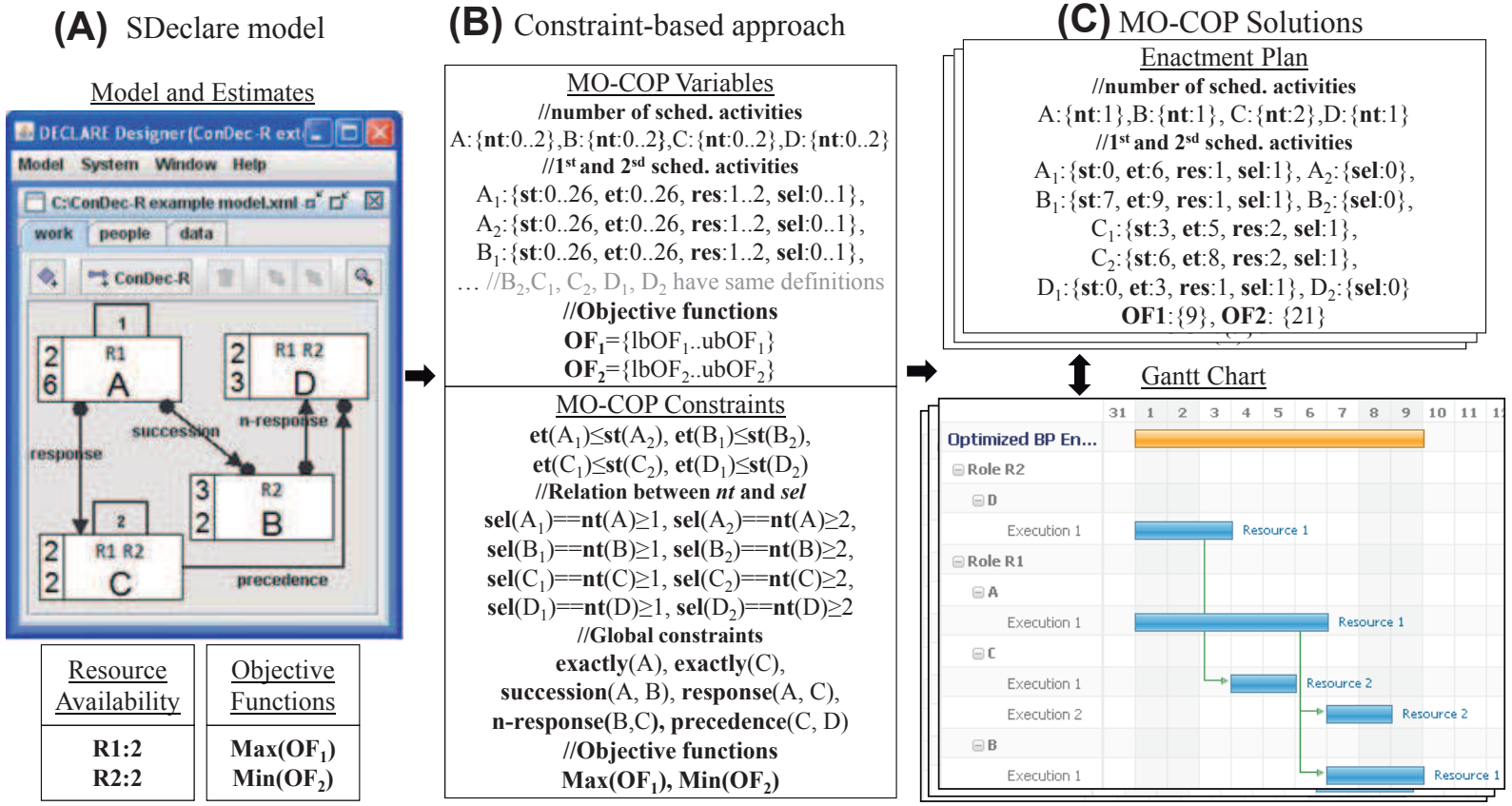

Figure 3.2: Generating Optimized Enactment Plans from SDeclare Models

3.1.3 Representing the SDeclare Model as a MO-COP Model

3.2 Generating Multi-Objective Optimized Enactment Plans

To generate optimal (or optimized) execution plans for a specific SDeclare model, the current Thesis proposes a constraint-based approach for P&S the BP activities. This includes: (1) the modelling of the problem as a MO-COP, (2) the use of global constraints implemented through filtering rules to improve the modelling of the problems and to efficiently handle the constraints in the search for solutions, and (3) a search algorithm for solving the MO-COP.

Given a process modeled as a SDeclare model (cf. Definition 25, Figure 3.2(A)), it needs to be represented as a MO-COP (cf. Definition 5, Figure 3.2(B)). Regarding the proposed MO-COP model, S-Activities (repeated activities in the MO-COP model, cf. Definition 26), which can be executed arbitrarily often if not restricted by any constraint, are modelled as a sequence of optional scheduling activities (cf. Definition 27). This is required since each execution of a S-Activity (i.e., a scheduling activity) is considered as one single activity which needs to be allocated to a specific resource and temporarily placed in the enactment plan, i.e., stating values for its start and end times.

Definition 26. *A **repeated activity** $ra = (a, Res, Att, nt)$ is a S-Activity $SAct = (a, Res, Atts)$ (cf. Definition 24) which can be executed several times. It defines a CSP variable which specifies the number of times the S-Activity is executed (i.e., nt).*

Definition 27. *A **scheduling activity** $sa = (st, et, res, sel)$ related to a repeated activity $ra = (a, Res, Att, nt)$, represents a specific execution of ra, where st and et are CSP variables indicating the start and the end times of the activity execution, respectively, $res \in Res$ is a CSP variable representing the resource used for the execution, and sel is a CSP variable indicating whether or not the activity is selected to be executed (i.e., equal to 0 in the case that it is not executed and equal to 1 otherwise).*

For each repeated activity, nt_{MAX}[4] scheduling activities exist, which are added to the CSP problem specification, apart from including a variable nt.

Moreover, additional CSP variables representing the objective functions to optimize are also included in the MO-COP (cf. Figure 3.2(B)). In this way, the SDeclare model $SDM = (SActs, Data, C_{BP}, AvRes, OFs)$ (cf. Definition 25) is transformed into a MO-COP P_o = *(V, D,* C_{CSP}*, OFs)* (cf. Definition 5, Figure 3.2(B)) where:

[4] nt_{MAX} represents the maximum value for the initial domain of nt (cf. Fig 2(B)).

1. $V = \{nt(a), a \in SActs\} \cup \{st(a_i), et(a_i), res(a_i), sel(a_i),\ i \in [1..\ nt_{MAX}(a)],\ a \in SActs\} \cup OFs$.

2. D is composed of the domains of each CSP variable *var*, where $UB(var)$ and $LB(var)$ represent the upper and lower bounds of the domain of *var*, respectively (cf. Example 10).

   Example 10. *In the example of Figure 2, the domain [0..2] is used for nt since 2 is the maximum cardinality for the BP activities (established by existence relations in the constraint-based model). The domain [0..26] is used for et and st since 26 would be the completion time if all the scheduling activities were serially executed taking the maximum cardinality for the BP activities into account.*

3. C_{CSP} is composed of the resource constraints, the global constraints (implemented by the filtering rules, cf. Section 3.2.1) related to C_{BP}, and the constraints which are inherent to the proposed model:

   (a) $\forall a \in SActs\ \forall i : 1 \leq i < nt(a) : et(a_i) \leq st(a_{i+1})$ (i.e., a specific execution of a repeated activity precedes the next execution of the same activity).

   (b) $\forall a \in SActs\ \forall i : 1 \leq i \leq UB(nt(a)) : sel(a_i) == nt(a) >= i$ (i.e., the *nt* variable of the repeated activity is directly related to the *sel* variables of the associated scheduling activities).

Resource constraints are not explicitly stated since most constraint-based systems provide a high-level constraint modeling specific to scheduling which includes an efficient management of shared resources. Besides the role-based allocation pattern, the CSP variables which are included in the model can be also used for specifying further resource constraints (Russell et al., 2005).

3.2.1 Global Constraints and Filtering Rules

Many constraint-based approaches for modelling and solving P&S problems have been proposed (Rossi et al., 2006). Moreover, several proposals exist for filtering rules related to specialized scheduling constraints (e.g., (Laborie et al., 2009; Bartak and Cepek, 2010)). Filtering rules lead to important performance improvements, facilitate the specification of the problem, and increase the efficiency in the search for solutions (Barba and Del Valle, 2011). Therefore, the considered problem could be managed by adapting existing constraint-based approaches.

```
TemporalPrecedence(A,B,[min, max]) ->
    If LB(nt(B)) > 0 then
        nt(A) <- nt(A) - {0}
    If LB(et(act(A,1))) + min > LB(st(act(B,1))))then
        LB(st(act(B,1)))) <- LB(et(act(A,1))) + min
    If UB(et(act(A,1))) - max > UB(st(act(B,1))) then
        UB(et(act(A,1))) <- UB(st(act(B,1))) - max
```

Figure 3.3: Propagator for Temporal Precedence Template in SDeclare

However, some SDeclare templates entail complex reasoning about several combined innovative aspects, such as the alternating executions of activities together with the varying number of times which these activities are executed. Therefore, specific global constraints have been implemented through innovative filtering rules to facilitate the specification of the problems and to increase the efficiency in the search for solutions. In this way, the constraints stated in the SDeclare specification (cf. Definition 25) are included in the MO-COP model through the related global constraints (cf. Figure 2(B)). In the MO-COP, initial estimates are made for upper and lower bounds of variable domains, and these values are refined during the search process by the developed filtering rules.

In this Thesis Dissertation, filtering rules related to the SDeclare constraints have been developed, i.e., choice, temporal and data constraints (cf. Example 11). In turns, the filtering rules associated the the basic Declare constraints were previous developed in (Barba and Del Valle, 2011). [5]

Example 11. *As an example, the* $TemporalPrecedence(A,B,[min,max])$ *rule is shown in Figure A, where the propagator that describes the pruning of domains appears after symbol* $\rightarrow$*. This constraint means that between min and max units of time before the first execution of B, at least one execution of A must be executed.*

3.2.2 Solving the MO-COP

Once the problem is modeled as a MO-COP (cf. Definition 5), several constraint-based mechanisms can be used to obtain the solutions to the MO-COP, i.e., multi-objective optimized enactment plans (cf. Definition 1). Since the generation of optimal plans presents NP-complexity (Garey and Johnson, 1979), it is not possible to ensure the optimality of the generated plans for all the cases. However, the developed constraint-based approach allows solving the considered problems in an efficient way as empirically demonstrate later in the case study.

The proposed constraint-based approach includes a multi-objective optimization search algorithm which is based on the ε-constraint method (Haimes et al.,

[5] A detailed description of the developed basic SDeclare filtering rules can be found at http://regula.lsi.us.es/MOPlanner/FilteringRules.pdf.

Algorithm 1: Generation of *EnactmentPlans* from a $MO-COP$

input : MO-COP P_0
output: Set<EnactmentPlan> *plans*

1 Map<ObjectiveFunction, Range> *ranges* = calculateRegions(P_0);
2 Set<Region> *regions*= divide(P_0, *ranges*);
3 Set<EnactmentPlan> *plans_with_dominated*= solveRegions(P_0, *regions*);
4 *plans*= removeParetoDominated(*plans_with_dominated*);

1971) (cf. Section 2.1). This search algorithm solves a number of single-objective COPs optimizing one of the objective functions and constraining the remaining objective functions. Specifically, given a MO-COP $P_o = (V, D, C_{CSP}, OFs)$ with N objective functions (i.e., $OFs = \{OF_1, .., OF_N\}$), the proposed algorithm (cf. Alg. 1) follows four steps:

1. For each objective function $OF_i \in OFs$, the related range (i.e., tentative maximum and minimum values that can be obtained for OF_i) is calculated (line 1 of Alg. 1) by using the algorithm `calculateRegions` (cf. Alg. 2). At the beginning of Alg. 2 an empty set of enactment plans is created for storing the solutions which are being generated (cf. line 1 of Alg. 2). Moreover, a solver which is in charge of finding solutions (i.e., enactment plans[6]) for single-objective COPs is created (i.e., *solver* at line 2 of Alg. 2). For each OF_i, a COP $P_i = (V, D, C_{CSP}, OF_i)$ which includes the same variables, domains and constraints than P_o but which only optimizes OF_i is generated (lines 4-6 of Alg. 2). Then, an incomplete complete search algorithm is used to find one optimized solution Sol_i for such problem within a given time limit (cf. line 7 of Alg. 2). The solution is then stored in the set *sols* (cf. line 8 of Alg. 2). All the solutions which are store in *sols* are then used to calculate a range of tentative maximum and minimum values for each objective function OF_i (cf. lines 9-13 of Alg. 2). This is performed by calculating the maximum and minimum values which are achieved for each OF_i in all the solutions stored in *sols* (cf. Example 12).

   Example 12. *For a MO-COP with three objective functions and* $sols = \{(OF_1 = 10, OF_2 = 5, OF_3 = 4), (OF_1 = 9, OF_2 = 6, OF_3 = 1)(OF_1 = 2, OF_2 = 4, OF_3 = 8)\}$*, the maximum (minimum) value for each* OF_i*, denoted as* OF_i^M *(*OF_i^m*), is:* $OF_1^M = 10, OF_2^M = 6$ *and* $OF_3^M = 8$ *(*$OF_1^m = 2, OF_2^m =$

[6] In the proposed approach the schedules (i.e., the raw solutions of the considered COPs) are directly transformed to enactment plans. Therefore, for the sake of simplicity, the solutions of such COPs are considered enactment plans.

4 *and* $OF_3^m = 1$.*). Then,* $range(OF_1) = [2,10]$, $range(OF_2) = [4,6]$ *and* $range(OF_3) = [1,8]$.

Algorithm 2: `calulateRegions` method: Calculate the range of values for each *ObjectiveFunction*.

input : MO-COP P_0
output: Map<ObjectiveFunction, Range> *ranges*

```
1  Set<EnactmentPlan> sols ← ∅;
2  COPSolver solver = createSolver();
3  foreach OF_i in P_0.OF_s do
4      COP P_i= createCOP();
5      P_i.⟨V,D,C_CSP⟩= P_0.⟨V,D,C_CSP⟩;
6      P_i.OF=OF_i;
7      EnactmentPlan Sol_i=solver.solve(P_i, Best, TIME_LIMIT);
8      sols.add(Sol_i);
9  foreach OFi in p0.OFs do
10     Range r=createRange();
11     r.max=maximumValue(sols, OF_i);
12     r.min=minimumValue(sols, OF_i);
13     ranges.put(OF_i, r);
```

2. With the goal of obtaining an uniformly distributed set of solutions for P_o, the solution space (i.e., a N-dimensional space) is divided into smaller N-dimensional regions (cf. line 2 of Alg. 1) by using the `divide` algorithm.[7] A region of a solution space with N objective functions consists of N sub-ranges, each one related to one objective function. In the `divide` algorithm, each range which is calculated for each objective function in the step 1, $range(OF_i)$, is divided into a given number DIV of non-overlapped sub-ranges, i.e., $range_j(OF_i)\ \forall j = 1 \ldots DIV$. Each sub-range $range_j(OF_i)$ of a range $range(OF_i)$ has the same size than the other sub-ranges related to the same objective function OF_i, with the exception of the first and the last sub-ranges, i.e., $\forall 2 \leq j \leq DIV - 1$: $range_j(OF_i) = [OF_i^m + (j-1) \times |range(OF_i)|/DIV, OF_i^m + j \times |range(OF_i)|/DIV]$, where $|range(OF_i)|$ refers to the size of $range(OF_i)$, i.e., OF_i^M - OF_i^m. Regarding the first and the last subranges, since the solution space is not totally explored in step 1 (since the search algorithm stops when a time limit is

[7] Due to its triviality, unlike the other algorithms, the `divide` algorithm is not formally shown in algorithm shape.

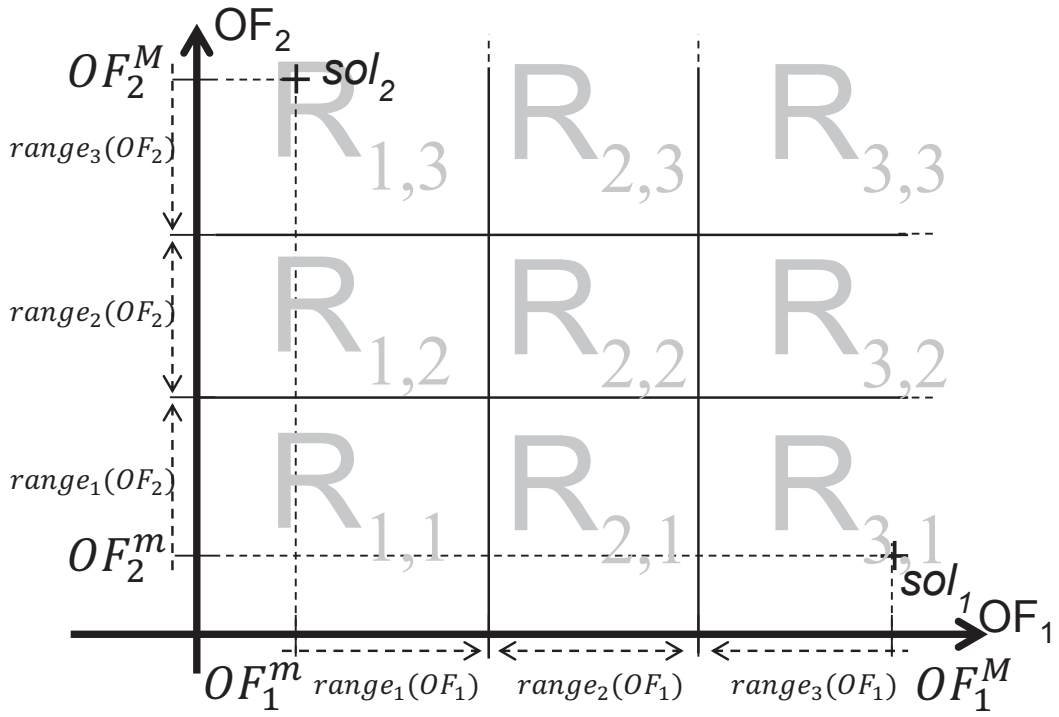

Figure 3.4: Solution space with two objective functions which is divided into nine regions.

reached) the absence of solutions out of the calculated ranges cannot be ensured. Therefore the minimum value of the first sub-range and the maximum value of the last sub-range are not fixed with the goal of avoiding missing some potential solutions. In this way, $range_1(OF_i) = [-\infty, OF_i^m + |range(OF_i)|/DIV]$, and $range_{DIV}(OF_i) = [OF_i^m + (DIV-1) \times |range(OF_i)| /DIV, +\infty]$. Then, the sub-ranges related to each objective function are combined with the sub-ranges related to all the other objective functions with the goal of obtaining different regions, R_v, where $v \in \mathbb{N}^N$ is a vector which contains the indices of the sub-ranges which belong to R_v, i.e, $range_j(OF_i) \in R_v$ if and only if $v[i] = j$, $\forall 1 \leq i \leq N$, $\forall 1 \leq j \leq DIV$ (cf. Example 13).

Example 13. *Figure 3.4 depicts a solution space which is divided in nine regions (i.e., $R_{1,1}$, $R_{1,2}$, .., $R_{3,3}$) for a MO-COP with two objective functions (i.e., OF_1 and OF_2) whose ranges are divided in tree sub-ranges (i.e., $range_1(OF_1)$, $range_2(OF_1)$, .., $range_3(OF_2)$).*

3. In order to look for a uniformly distributed set of solutions, each region is independently managed (line 3 of Alg. 1) by using the algorithm `solveRegions` (cf. Alg. 3).

Algorithm 3: `solveRegions` method: Generate *EnactmentPlans* by solving the *COPs* related to each *region*.

input : MO-COP P_0
Set<Region> *regions*
output: Set<EnactmentPlans> *plans*

```
1  Set<Region> dominated ← ∅;
2  foreach region in regions by region.countDominatingRegions(regions)
   desc do
3      if !dominated.contains(region) then
4          Set<EnactmentPlan> sols ← ∅ ;
5          foreach OF_i in P_0.OFs do
6              COP P_{v,i}= createCOP();
7              P_{v,i}.⟨V,D,C_CSP⟩ = P_0.⟨V,D,C_CSP⟩;
8              P_{v,i}.OF=OF_i;
9              foreach OF_l in P_0.OFs do
10                 P_{v,i}.C_CSP.add("OF_l ∈ region.get(OF_l).getRange()");
11             sols =solver.solve(P_{v,i}, Anytime, TIME_LIMIT);
12         if !sols.isEmpty() then
13             dominated.addAll(calculateDominatedRegions(regions,
               region));
14             plans.addAll(removeParetoDominatedSolutions(sols));
```

Initially, an empty set of dominated regions is created (cf. line 1 of Alg. 3). This set is created with the goal of storing all the regions which are dominated by others. Since only the Pareto optimized solutions are considered, the order of solving the regions influences the efficiency since some calculus can be saved by applying a proper ordering. Thus, this algorithm solves the aforementioned problems $P_{v,i}$ (i.e., a problem which optimizes the objective function i in the region R_v) starting with those problems which belong to the region which dominates more regions (line 2 of Alg. 3). A region R_v dominates $R_{v'}$ if and only if $\forall 1 \leq k \leq N : v[k] > v'[k]$ (being N the number of dimensions), cf. Example 14.[8] If a solution is found in a region, all the COPs related to the regions which are dominated by the former do not need to be solved (line 3 of Alg. 3) since all their solutions are Pareto dominated by any solution which belongs to the former region.

[8]For the sake of clarity, the maximization of each objective function is assumed. The problem of minimization is analogous.

Example 14. *In Figure 3.4, region $R_{3,3}$ dominates $R_{2,2}, R_{2,1}, R_{1,2}$ and $R_{1,1}$, region $R_{2,3}$ dominates $R_{1,2}$ and $R_{1,1}$, and so on.*

Therefore, for each region R_v which is not dominated, an empty set of solutions is created (cf. line 4 of Alg. 3) to store all the enactment plans related to such region. In addition, N COPs $P_{v,i} = (V, D, C_v, OF_i)$ are generated (cf. lines 5 and 6 of Alg. 3), where $C_v = C_{CSP} \cup (OF_l \in range_{v[l]}(OF_l), \forall 1 \leq l \leq N)$ (i.e., which only optimizes OF_i and where all the objectives functions are constrained to be in the related sub-range $range_{v[l]}(OF_l)$; cf. lines 7-10 of Alg. 3). Unlike step 1, in order to generate a wide set of solutions, an *anytime optimization algorithm* (Zilberstein, 1996) is used which is an incomplete search algorithm which updates the best solution which is found during the search. Then not only the best solution (cf. line 11 of Alg. 3) but some intermediate solutions are returned.

If at least one solution is found within a region R_v (cf. line 12 of Alg. 3), the regions which are dominated by R_v are included in the set *dominated* to avoid the search for solutions in that dominated region (cf. line 13 of Alg. 3). In addition, the solutions which are obtained within R_v are filtered by removing the solutions which are Pareto dominated (cf. line 14 of Alg. 3 and Example 15).[9]

Example 15. *For a MO-COP with two objective functions and a solution space divided in nine regions, Figure 3.5 shows the different solutions obtained within each region, where no solutions are found in regions $R_{2,3}$ and $R_{3,3}$, and where $R_{1,1}$ and $R_{2,1}$ are eliminated (i.e., the COPs related to them are not solved) since some solutions are found in region $R_{3,2}$ which dominates $R_{1,1}$ and $R_{2,1}$. In Figure 3.5, each cross represents a solution which is Pareto dominated by another solution in the same region.*

4. After all the COPs are solved (i.e., a diversified set of solutions is obtained), solutions which are dominated by solutions from a different region are removed (cf. line 4 of Alg. 1). Then a distributed set of Pareto optimized solutions is obtained (cf. Example 16).[10]

Example 16. *In Figure 3.5 all solutions which are dominated by any solution which belongs to a different region are depicted by a cross inside a box, and all the Pareto optimized solutions are depicted by a square.*

[9] In a general case, the complexity of the Pareto dominance algorithm is $O(n^2)$ where n is the number of solutions [referecia]. Then, the fact of having all the solutions divided in non-overlapped regions (i.e., the solutions are clustered) reduces the complexity since $O(n^2) < O((n/m)^2) \times m, \forall m > 1$ where m is the number of regions.

[10] The complete set of Pareto optimal solution is not the goal of the proposed algorithm, but a representative and distributed set of Pareto optimized solutions.

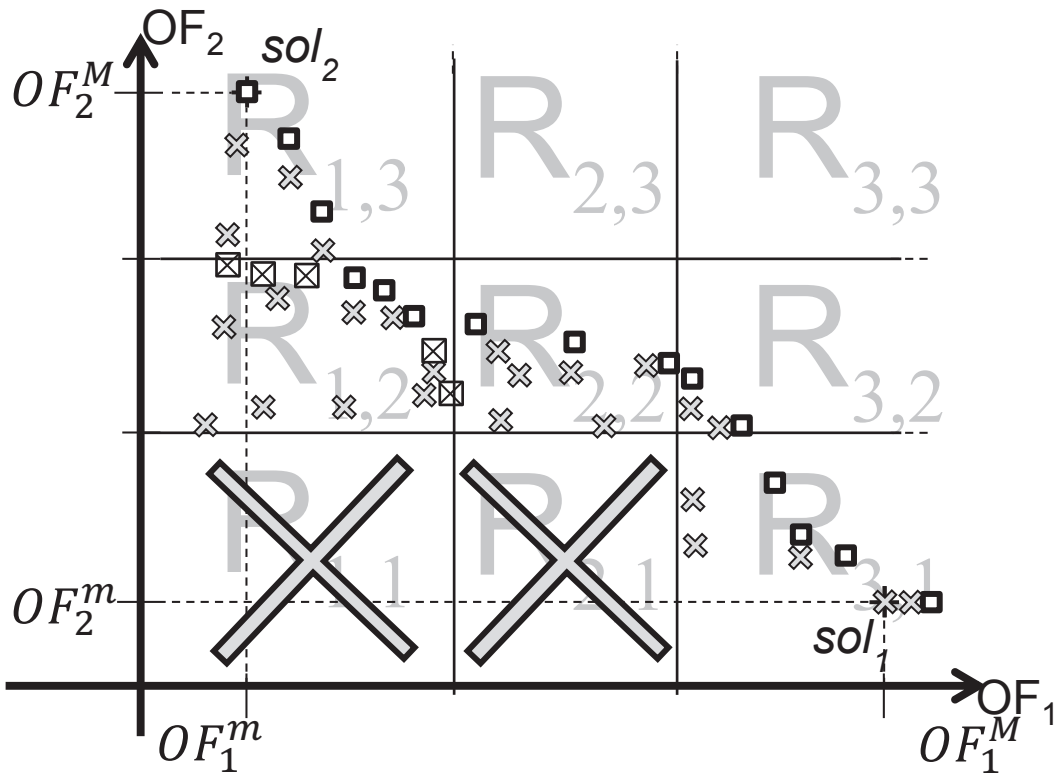

Figure 3.5: Set of solutions for a 2-objectives MO-COP where the Pareto optimized solutions are depicted by squares, solutions which are Pareto dominated by solutions from the same region are depicted by crosses, and solutions which are Pareto dominated by solutions from other regions are depicted by crosses inside a box. Dominated regions are indicated with a big cross.

3.3 Other applications of the Optimized BP Enactment Plans

3.3.1 User Recommendations for the Optimized Execution of BPs

Introduction

In order to support the users during process execution in optimizing performance goals (e.g., minimizing the overall completion time), the generation of optimized enactment plans was proposed (cf. Section 3.2). Recommendations on possible next steps are then generated taking the partial trace and the optimized plans into account. Replanning is supported if actual traces deviate from the optimized enactment plans (e.g., because estimates turned out to be inaccurate).

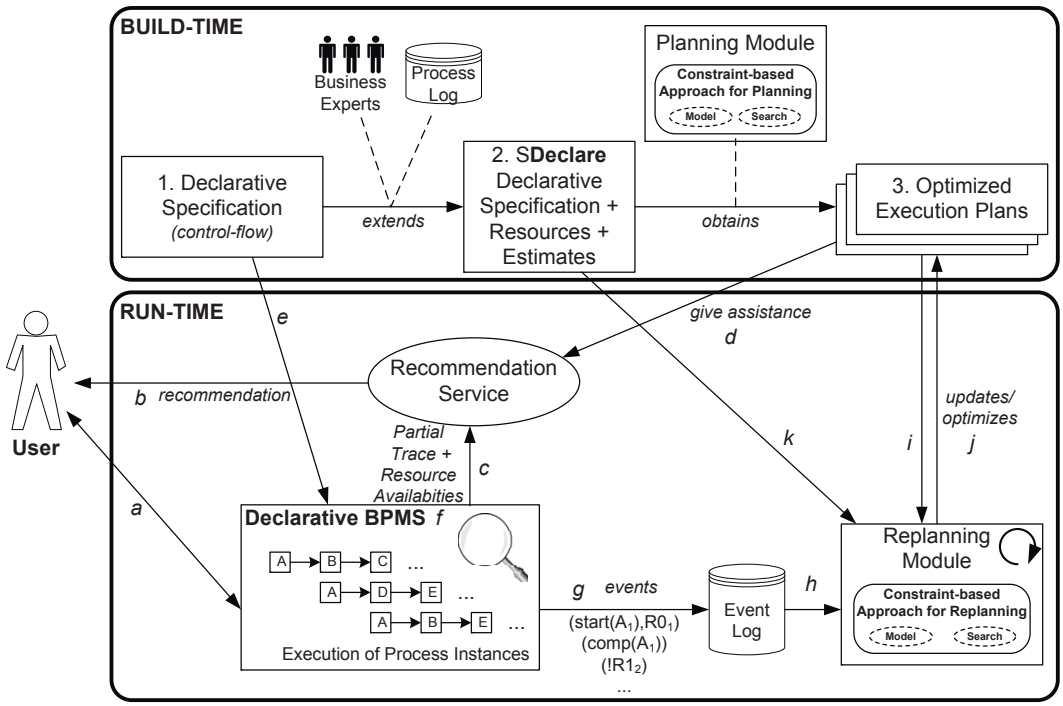

Figure 3.6: Overview of the proposal for generating optimized execution plans at build-time and generating recommendations at run-time.

Generating Recommendations on Possible Next Execution Steps

As stated, constraint-based processes offer much flexibility. Typically, given a constraint-based process model and a certain partial trace, users can choose from several enabled activities which activity to execute next, which is a challenging selection in most cases. In order to address this challenge this section proposes an approach to assist users during process execution in optimizing performance goals like minimizing the overall completion time. Specifically, users are supported during process execution by a recommendation service which provides recommendations on how to proceed best with the execution. Hereby, a recommendation (cf. Definition 28) is composed by one or more enabled activities (cf. Definition 19) to be executed next, together with their resource allocations since both control-flow and resource perspectives are considered.

Definition 28. *A **recommendation** Rec is composed by a set of pairs* (a_i, Rj_k) *suggesting to start the i-th execution of activity a using resource* Rj_k[11].

[11] Rj_k refers to the k-th resource with role j.

For example, the recommendation $< (A_1, R0_1), (B_2, R1_2) >$ suggests to start the first execution of activity A using resource $R0_1$ and the second execution of activity B using resource $R1_2$.

The recommendation service is based on optimized enactment plans which are already generated during build-time (cf. Section 3.2) by P&S all BP activities and further optimized during run-time. At specific times of the process execution, the recommendation system generates the recommendations by considering: (1) the optimized enactment plans, (2) the partial traces (cf. Definition 17) of the process instances to be optimized, and (3) the resource availabilities. In order to determine the recommendations, different strategies can be used (which will be described later). Thereby, the recommendation service ensures that not only single process instances get optimized, but the whole set of instances which is planned to be executed within a certain timeframe, hence allowing for a global optimization.

At run-time, process instances (cf. Definition 18) are executed by authorized users (a in Figure 3.6). At any point during the execution of a process instance, the user can select from the set of enabled activities (cf. Definition 19) what to do next. However, to guide the user to optimize the overall process goals, recommendations (cf. Definition 28) are provided by the recommendation service (b in Figure 3.6). Note that the user is not obliged to follow the recommendations but she can select any of the enabled activities, i.e., all the flexibility of the declarative specification is kept. To provide recommendations, the recommendation service proposes the most suitable activity to execute next, i.e., proposes the recommendation with the highest quality. [12]

Algorithm 4 shows how the recommendations are generated. As input data some information is required: (1) the SDeclare specification of the problem (cf. Definition 25) and (2) the initial optimized enactment plans (cf. Definition 1) generated during the build-time phase. As stated, for a particular timeframe a BP enactment plan for a set of instances (cf. Definition 18) is generated. Algorithm 4 starts at the beginning of such a timeframe and lasts until all the planned instances have completed (line 15 in Alg. 4).

Algorithm 4 continuously generates recommendations (line 11) on how to proceed with process execution considering (1) the best available enactment plan (d in Figure 3.6) meeting the constraints imposed by the constraint-based specification (e in Figure 3.6), and (2) all events that occurred during process execution (i.e., *allEvents*). This includes (1) the current partial traces of the process instances (c in Figure 3.6), and (2) the current information about resource availabilities (c in Figure 3.6), e.g., $(!Rj_k, T)$ means that k-th resource with role j becomes un-

[12] As multi-objective optimization is considered in this Thesis, the value of the quality is calculated from the values of the objective functions.

Algorithm 4: Provide Recommendations

input : SDeclare Specification *cr*
set<EnactmentPlan> *plans*

```
1  Recommendation rec;
2  Set<Event> allEvents ← ∅;
3  Set<Event> newEvents;
4  int T ← currentTime();
5  repeat
6      if event(newEvents, T) then
7          allEvents ← allEvents ∪ newEvents;
8          plan ← update(cr, plans, allEvents);
9      if optimizerPlan(cr, plans, allEvents)! = null then
10         plan ← optimizerPlan(cr, plans, allEvents);
11     rec ← generateRecommendation(plans, allEvents);
12     if rec! = null then
13         send(rec);
14     T ← currentTime();
15 until !CompleteTrace(cr, allEvents);
```

available at time T. In the case that a recommendation is suggested (line 12), the recommendation system sends it to the user (line 13).

As execution proceeds, the BP enactment and the resource availabilities are monitored (f in Figure 3.6). If there are new events at time T (line 6 in Alg. 4), i.e., activities get started/completed or resources become available/unavailable (g in Figure 3.6), then the set of events *allEvents*, which includes both the partial trace and the resource availability events, is updated (line 7 in Alg. 4). By doing this, the proposed approach is able to deal with uncertainty involved (i.e., inaccurate estimates, unexpected changes in resource availabilities, and user deviations).

Whenever events are updated the Replanning Module (h in Figure 3.6) analyzes the optimized plans (i in Figure 3.6) as well as the events. In particular, it checks if the current execution traces match with any of the optimized enactment plans (and if a recommendation can be made) or whether updates of the execution plans are needed (j in Figure 3.6). In general, updates of the execution plan can become necessary due to deviations (line 8 in Alg. 4), i.e., (1) the execution trace is not part of one of the optimized enactment plans (e.g., the user is not always following the recommendations), (2) estimates are incorrect (e.g., when activity executions take longer/shorter than estimated, or more or less instances than ex-

pected get executed), or (3) resource availabilities change (i.e., resources become unavailable).

Moreover, the Replanning Module is continuously searching for a better plan by considering the event log during BP execution, provided that the current plan is not optimal. In this way, plan updates are conducted whenever the replanning module finds a solution which is better than the current optimized plans (lines 9 and 10 in Alg. 4). If plan updates are required, the Replanning Module needs to access the extended constraint-based specification of the process (k in Figure 3.6) to generate new optimized plans considering both the estimates and the constraint-based specification. If necessary, the replanning, i.e., the generation of new optimized enactment plans, is carried out by applying a constraint-based approach for P&S the BP activities (cf. Section 3.2).

Despite the NP-complexity of the considered problems, in general 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.

3.3.2 Automatic Generation of Optimized Imperative BP Models

Introduction

To support process analysts in the definition of optimized BP models a method for automatically generating imperative BP models using AI planning techniques from constraint-based specifications is suggested. In the proposed approach, the static part of the input declarative model (i.e., control-flow and resource constraints) is expected to be useful on a long-term basis since it embraces information which is not supposed to change often. The base declarative model (i.e., only including the static part) is complemented with information that is less stable and which is potentially unknown until starting the BP execution. From this extended model, the proposed approach is in charge of determining how to satisfy the constraints imposed by the declarative specification and at the same time to attain an optimization of certain objective functions (e.g., minimization of completion time). For this optimization, scheduling is done on a short-term basis by considering the optimization of a set of instances.

Unlike conventional proposals, in this approach each generated model is created and deployed for a specific planning period, considering changing information such as the number of process instances which are being executed within a specific timeframe. For the next executions of the declarative model, new models will be generated considering the specific values which are given for the changing

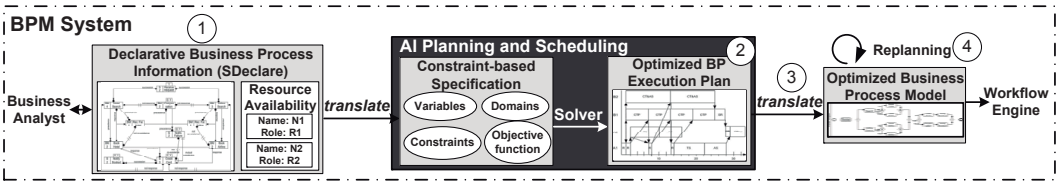

Figure 3.7: AI P&S techniques for the generation of optimized BP models.

information. Since planning is done on a short-term basis, the generated models are less prone to change.

Figure 3.7 provides an overview of this section. Taking the constraint-based specifications as a starting point (cf. Figure 3.7(1)), enactment plans can automatically be generated (cf. Figure 3.7(2), Section 3.2).

The generated enactment plans are then automatically translated into a BPMN model (BPMN, 2011) (cf. Figure 3.7(3)), which can be then further improved by a business analyst, where necessary. In most cases, BPMN models can be translated into an execution language (Ouyang et al., 2006), such as BPEL (BPEL, 2007), which enables BP designs to be deployed into BPMSs and let their instances be executed by a BPM engine. To provide for an increased flexibility the BPMN model can be dynamically adapted during run-time by using replanning (cf. Figure 3.7(4)).

Note that the BPMN model is generated with the goal of making the declarative model automatically executable by a BPMS by considering the specific values of the changing information which are given just before starting the execution the process. In this way, application of decision deferral patterns is automated (Reichert and Weber, 2012), i.e., the role of the BPMS is rather focused on enabling control and ensuring compliance (decisions are automatically made by the BPMS). Regarding decision deferral patterns, this approach belongs to the late modelling and composition pattern, i.e., allowing for modelling and automatic composition of a process model just before starting the execution of a branch of process instances. Therefore, this approach can be framed within dynamic process-based composition (i.e., completely creating the executable process model dynamically at run-time), which constitutes an example of the automated variant of the late modelling and composition pattern.

In this way, the automatic generation of BP models simplifies the BP design phase by facilitating the human work in most cases, preventing failures in the developed BP models, and enabling better optimization to be attained in the enactment phase. Furthermore, imperative BP models can dynamically be generated from static constraint-based specifications just before starting the BP enactment, once some values for the enactment parameters, e.g., resource availabilities, are

known. Moreover, the automatic generation of BP models can deal with complex problems of great size in a simple way. Therefore, a wide study of several aspects can be carried out, such as those related to the requirement of resources of different roles, or the estimated completion time for the BP enactment, by generating several kinds of alternative specifications. In addition, in order to address run-time flexibility the proposed approach allows decisions to be deferred at run-time by using complex late-planning activities, and the BPMN model to be dynamically adapted during run-time using replanning.

From Optimized Enactment Plans to Optimized Business Process Models

Section 3.2 has described how optimized BP enactment plans can be generated from SDeclare specifications. This section describes how a BPMN model which includes the same activities to be executed in the same ordering and also using the same resources can be generated from the optimized enactment plan.

For each role in the BP enactment plan, a BPMN pool (cf. Definition 29) is created, which contains as many lanes as number of available resources for that role.

Definition 29. *A **BPMN pool** $BPMNPool = (role, \#role)$ is a pool of a BPMN model, which is composed of $\#role$ lanes.*

Moreover, for each scheduling activity in the BP enactment plan a BPMN activity (cf. Definition 30) is created. Additionally, one start activity and one end activity are included in the BPMN model.

Definition 30. *A **BPMN activity** $BPMNAct = (pool, lane, dur, st)$ is an activity of a BPMN model placed in the lane named $lane$ of the pool named $pool$, with duration dur and start time st.*

One of the most important aspects to be considered for the generation of optimized BPMN models are the precedence relations between the BPMN activities (scheduling activities). For establishing these precedence relations the values for the start and the end times of the scheduling activities in the enactment plan are considered. These precedence relations are then used as a basis for generating BPMN models (cf. Definition 34) from BP enactment plans. Some related definitions are given below:

Definition 31. *In a BP enactment plan regarding a CSP solution S, a scheduling activity a_i is a **predecessor** of another scheduling activity b_j, $a_i \in predecessors(b_j)$, if the relation $S^{et(a_i)} \leq S^{st(b_j)}$ holds due to resource or template relations.*

Definition 32. *In a BP enactment plan, a scheduling activity a_i is a **direct predecessor** of another scheduling activity b_j, $a_i \in DP(b_j)$, if $a_i \in predecessors(b_j) \wedge \nexists c_k \in predecessors(b_j) \mid a_i \in predecessors(c_k)$.*

Definition 33. *In a BP enactment plan, a scheduling activity a_i is an **indirect predecessor** of another scheduling activity b_j, $a_i \in IP(b_j)$, if $a_i \in predecessors(b_j) \wedge \exists c_k \in predecessors(b_j) \mid a_i \in predecessors(c_k)$.*

Definition 34. *A **BPMN model** $BPMN = (Pools, Activities, SequenceFlows, ParallelM)$ related to a SDeclare process model $CR = (Acts, Data, C_{BP}, Res, OFs)$ (cf. Definition 25) and to a solution S (cf. Definition 3) of the related CSP (cf. Definition 2) is a BP model specified through the BPMN language, where:*

1. $Pools = \{BPMNPool(role, \#role), (role, \#role) \in Res\}$.
2. $Activities = \{BPMNAct(role(a), S^{res(a_i)}, dur(a), S^{st(a_i)}), (a, role, dur) \in Acts, i \in [1..S^{nt(a)}]\} \cup \{start = BPMNAct(P_0, L_0, 0, 0)\} \cup \{end = BPMNAct(P_0, L_0, 0, max_{(a,role,dur)\in Acts, i\in[1..S^{nt(a)}]} S^{et(a_i)})\}$.
3. *Let the set Predecessors be:*

   I $\{(start, a_i) \mid (a, role, dur) \in Acts, i \in [1..S^{nt(a)}], S^{st(a_i)} = 0\} \cup$

   II $\{(a_{S^{nt(a)}}, end) \mid (a, role, dur) \in Acts, \nexists b_i, i \in [1..S^{nt(b)}], (b, role_b, dur_b) \in Acts, a_{S^{nt(a)}} \in predecessors(b_i)\} \cup$

   III $\{(b_i, c_j) \mid i \in [1..S^{nt(b)}], (b, role_b, dur_b) \in Acts, j \in [1..S^{nt(c)}], (c, role_c, dur_c) \in Acts, b_i \in DP(c_j)\}$,

   Then:

   (a) $SequenceFlows = \{(b_i, c_j) \mid (((b, role_b, dur_b) \in Acts \wedge i \in [1..S^{nt(b)}]) \vee b_i = start) \wedge (((c, role_c, dur_c) \in Acts \wedge j \in [1..S^{nt(c)}]) \vee c_j = end) \wedge (b_i, c_j) \in Predecessors \wedge |\{d_k, (((d, role_d, dur_d) \in Acts \wedge k \in [1..S^{nt(d)}]) \vee d_k = start), (d_k, c_j) \in Predecessors\}| = 1)\}$.

   (b) $ParallelM = \{(Sources, c_j) \mid (((c, role_c, dur_c) \in Acts \wedge j \in [1..S^{nt(c)}]) \vee c_j = end) \wedge Sources = \{b_i, (((b, role_b, dur_b) \in Acts \wedge i \in [1..nt(b)]) \vee b_i = start) \wedge (b_i, c_j) \in Predecessors\} \wedge |Sources| > 1\}$.

In this way, through the set *Predecessors*, the precedence relations between activities are stated so that (1) the start activity is predecessor of all scheduling activities whose *st* value is equal to 0, (2) the activities which are not predecessors of any other activity, are predecessor of the end activity, and (3) in general, one

activity b_i is predecessor of another activity c_j iff b_i is direct predecessor of c_j (cf. (3) in Definition 34) . The set *Predecessors* is represented in the BPMN model by BPMN sequence flows between a source activity b_i and a sink activity c_j, in the case that b_i is the only predecessor of c_j (cf. (3)(a) in Definition 34), or by a parallel merging gateway between a set of source activities *Sources* and a sink activity c_j in the case that c_j has more than one predecessor (cf. (3)(b) in Definition 34). Note that parallel merging gateways (i.e., parallel gateways which have several sources and only one sink) need to be explicitly included in the resulting BPMN model, since they do not have the same meaning as several binary sequence flows from several sources and one sink. However, parallel splitting gateways (i.e., parallel gateways which have several sinks and only one source) do not need to be explicitly included in the resulting BPMN model since several binary sequence flows between one source activity and several sink activities have the same meaning as a parallel splitting gateway in the BPMN language.

In order to develop the algorithms to generate the BP models from the optimized enactment plans, certain related types are stated, as shown in Figure 3.8 (UML diagram). Note that at this point of the process the CSP variables are instantiated, and hence all the information is known (`nt` variable for each BP activity, `st` variable for each scheduling activity, resource in which each scheduling activity is executed, etc). The types which appear in the UML diagram are as follows:

- $OptimizedPlan(acts, r, t)$: This represents the generated optimized enactment plan. Moreover, it contains the information related to the input problem. Specifically, this type contains properties regarding a set of roles r, a set of repeated activities (SDeclare activities) $acts$, and a set of constraints which relate the repeated activities t.

- $RepeatedAct(role, dur, acts, nt)$: This represents the SDeclare activities. Each repeated activity contains information about the required role (i.e., $role$), the estimated duration (i.e., dur), the set of scheduling activities which represent the execution of each BP activity (i.e., $acts$), and the number of times this repeated activity is executed (i.e., nt).

- $Role(resources)$: This represents a role, and it is composed of the set of resources available for this role.

- $Resource(acts)$: This represents a resource. This type contains properties regarding a list of scheduling activities which are executed in that resource, ordered by the start time.

- $Constraint(name)$: This represents the high-level relations which are given between the repeated activities. Two specializations are included to allow

the relations between one source and several sinks (ConstraintSinks), and between several sources and one sink (ConstraintSources)[13]. The method *includePred* of a template updates the information of the BPMN model by including the precedence relations which are implied by that template (more details are given later in this section during the presentation of the algorithms). For the generation of the BPMN model, the constraints are considered for the connection of the BPMN activities.

- $P\&SAct(st, et, res)$: This represents each execution of a repeated activity. This type contains properties regarding the start and the end times of the activity, together with the resource used by the scheduling activity (*st*, *et* and *res* respectively). Since each P&SAct is related to a specific BPMNAct in the resulting BPMN model, the P&SAct type provides the method *toBPMNAct* in order to obtain the related BPMNAct from a P&SAct (cf. Figure 3.8). This method is formalized as follows, where the symbol $\rightarrow$ is used to specify the output parameter: $toBPMNAct(a : P\&SAct) \rightarrow BPMNAct(a.res.lane.pool, a.res.lane, a.dur, a.st)$.

- $BPMNModel(pools, acts, seqFlows, gates)$: This represents the BPMN model that is generated. This model is composed of a set of pools *pools*, a set of BPMN activities *acts*, a set of sequence flows *seqFlows*, and a set of gates *gates*. It contains the function $createBPMN() \rightarrow BPMNModel(\emptyset, \emptyset, \emptyset, \emptyset)$ (i.e., this method returns an object of type BPMNModel in which all properties are empty sets).

- $BPMNAct(pool, lane, dur, st)$: This represents a BPMN activity. This type contains properties regarding the pool and the lane where the activity is allocated (i.e., *pool* and *lane* respectively), together with the estimated duration *dur* and start time *st*. It contains the following functions (cf. Figure 3.8): (1) $createBPMNAct(a : P\&SAct) \rightarrow BPMNAct(a.res.lane.pool, a.res.lane, a.dur, a.st)$, which creates a BPMNAct from a P&SAct, and (2) $createBPMNAct(p : Pool, l : Lane, dur : int, st : int) \rightarrow BPMNAct(p, l, dur, st)$.

- $Pool(lanes, role)$: This represents a BPMN pool. Each pool is associated to a specific object of type Role *role*, and is composed of a set of objects of type Lane *lanes*. It contains the function $createPool(role : Role) \rightarrow Pool(lanes, role)$, where $lanes = \bigcup_{res \in role.resources} createLane(res)$, i.e., for each resource of that role, a related lane is created and included in the pool.

[13] Note that both ConstraintSinks and ConstraintSources can be used for specifying binary constraints.

- $Lane(res)$: This represents a BPMN lane. Each lane is associated to a specific resource *res*. It contains the function $createLane(res : Resource) \rightarrow Lane(res)$.

- *Gate*: This represents a BPMN gate. In order to consider parallel merging gateways, a specialization, named ParallelM, is developed.

- $ParallelM(sources, sink)$: This represents a parallel merging gateway, together with the related input and output connections of the gateway. This type contains properties regarding a set of inputs *sources*, and one output *sink*. It contains the function $createParallelM(l : Set < BPMNAct >, a : BPMNAct) \rightarrow ParallelM(l, a)$.

- $SequenceFlow(a, b)$: This represents a precedence sequence flow between two BPMN activities, *a* and *b*. It contains the function $createSequenceFlow($ $a : BPMNAct,\ b : BPMNAct) \rightarrow SequenceFlow(a, b)$. Note that the connections between a BPMN activity and a gateway are stated in ParallelM objects.

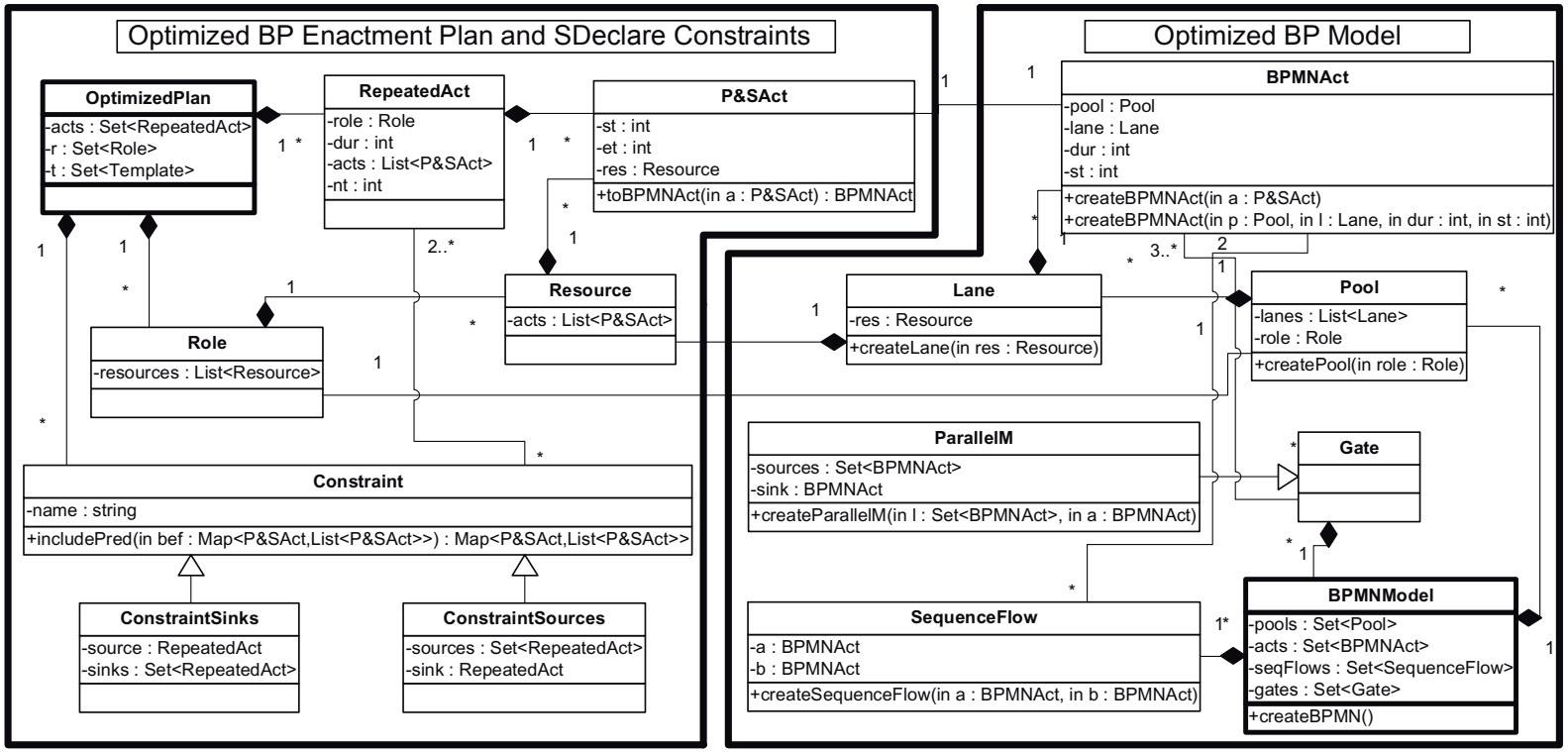

Figure 3.8: UML Diagram of Types for the Optimized BPMN Generation.

Algorithm 5: Construct an Optimized BP Model from an Optimized BP Enactment Plan

input : SortedSet <P&SAct> *acts*, ordered by *st*
Set<Constraint> *c*
Set<Role> *r*
output: BPMNModel *bp*

1 $bp \leftarrow createBPMN()$;
2 $bp.pools \leftarrow \{createPool(role) \mid role \in r\}$;
3 $bp.acts \leftarrow \{createBPMNAct(a) \mid a \in acts\}$;
4 BPMNAct *start* $\leftarrow createBPMNAct(P_0, L_0, 0, 0)$;
5 BPMNAct *end* $\leftarrow createBPMNAct(P_0, L_0, 0, max_{a \in acts} a.et)$;
6 $bp.sequenceFlows \leftarrow \{createSequenceFlow(start, ini) \mid ini \in bp.acts, ini.st = 0\}$;
7 Map<P&SAct,Set<P&SAct >> *pred* $\leftarrow CreateDependencies(acts, c, r)$;
8 **foreach** *psact in acts* **do**
9 **if** *pred(psact).size == 1* **then**
10 P&SAct *aPred* $\leftarrow pred(psact).get(0)$;
11 $bp.sequenceFlows \leftarrow bp.sequenceFlows \cup createSequenceFlow(toBPMNAct(aPred), toBPMNAct(psact))$;
12 **else**
13 Set<BPMNAct> *inputs* $\leftarrow \{toBPMNAct(a) \mid a \in pred(psact)\}$;
14 $bp.gates \leftarrow bp.gates \cup createParallelM(inputs, toBPMNAct(psact))$;
15 Set<BPMNAct> $finals \leftarrow \{toBPMNAct(a) \mid a \in P\&SAct, \neg\exists b \in P\&SAct, a \in pred(b)\}$;
16 **if** *finals.size == 1* **then**
17 BPMNAct $final \leftarrow finals.get(0)$;
18 $bp.sequenceFlows \leftarrow bp.sequenceFlows \cup createSequenceFlow(final, end)$;
19 **else**
20 $bp.gates \leftarrow bp.gates \cup createParallelM(finals, end)$;
21 **return** *bp*;

In Algs. 5, 6, 7, 8, T<P> represents the generic type T with the generic parameter instantiated to P. These algorithms are explained below.

The main algorithm, Alg. 5, constructs a BPMN model from an optimized BP enactment plan (cf. Definition 1) and a SDeclare model (cf. Definition refdefsdeclareprocessmodel). From the enactment plan and the SDeclare model, the input

parameters of Alg. 5 can be stated, i.e., a sorted set of scheduling activities ordered by start time (i.e., *acts*); a set of the constraints which relate the repeated activities (i.e., *c*); and a set of the considered roles (i.e., *r*). Algorithm 5 starts by creating an empty BPMN model (cf. line 1). Moreover, a pool associated to each role is created, together with the corresponding lanes (line 2). In a similar way, a BPMN activity associated to each scheduling activity is created (line 3). The start and end activities of the model can be associated to any pool, which is represented by P_0 in Alg. 5, and to any lane, which is represented by L_0 in Alg. 5 (lines 4 and 5 respectively). In line 6, a sequence flow between the start BPMN activity and each BPMN activity whose estimated start time is equal to 0 is created through the *createSequenceFlow* method (cf. Figure 3.8). As explained, the *createSequenceFlow* method contains the parameters (1) *a* of type *BPMNAct*, and (2) *b* of type *BPMNAct* as input, and creates a *SequenceFlow* object which states a BPMN binary precedence relation starting in *a* and ending in *b*. After that, the map *pred* associates a set of direct predecessors (cf. Definition 32) to each scheduling activity by using the method *CreateDependencies* (cf. Alg. 6, explained later in this section) in order to generate the BPMN model (line 7) [14].

Lines 8-14 establish the sequence flows and gateways between the BPMN activities in the following way: if the BPMN activity has only one direct predecessor, a sequence flow is included (lines 9-11); otherwise if the BPMN activity has several direct predecessors, a parallel merging gateway is included through the *createParallelM* method (lines 12-14). As explained, the *createParallelM* method contains the parameters (1) *l* of type $List < BPMNAct >$, and (2) *a* of type *BPMNAct* as input, and creates a *ParallelM* object which states a BPMN parallel merging gateway (also including all the related connections) with contains all the BPMN activities of *l* as input and the BPMN activity *a* as output. In line 15, all the final activities are selected to be direct predecessors of the end activity. These activities are related by either a sequence flow, in the case that there is only one ending activity (lines 16-18); or by a parallel merging gateway, in the case that there are several ending activities (lines 19-20). Note that, as mentioned, parallel merging gateways (i.e., parallel gateways which have several sources and only one sink) need to be explicitly included in the resulting BPMN model, since they do not have the same meaning as several binary sequence flows from several sources and one sink. However, parallel splitting gateways (i.e., parallel gateways which have several sinks and only one source) do not need to be explicitly included in the resulting BPMN model since several binary sequence flows between one source activity and several sink activities have the same meaning as a parallel splitting gateway in the BPMN language.

[14] The generic type $Map < T1, T2 >$, which associates an object of type $T2$ to an object of type $T1$, is used.

Algorithm 6: CreateDependencies

input : SortedSet<P&SAct> *acts* ordered by *st*
Set<Constraint> *constraints*
Set<Role> *roles*
output: Map<P&SAct,Set<P&SAct >> *directPredecessors*

```
1  Map<P&SAct,Set<P&SAct >> allPredecessors ← ∅;
2  foreach r in roles do
3      foreach res in r.resources do
4          List<P&SAct> actsRes ← res.acts;
5          foreach i in i:1..actsRes.size-1 do
6              allPredecessors(actsRes_{i+1}) ← {actsRes_i};
7  foreach c in constraints do
8      c.includePredecessors(allPredecessors);
9  Map<P&SAct,Set<P&SAct >> indirectPredecessors ← ∅;
10 foreach act in acts do
11     directPredecessors(act) ← allPredecessors(act);
12     foreach p in allPredecessors(act) do
13         directPredecessors(act) ←
           directPredecessors(act) \ allPredecessors(p);
14         indirectPredecessors(act) ←
           indirectPredecessors(act) ∪ allPredecessors(p);
15     allPredecessors(act) ←
       allPredecessors(act) ∪ indirectPredecessors(act);
16 return directPredecessors;
```

As stated before, one of the most important aspects to be considered for this model generation are the precedence relations between the scheduling activities of the plan, which are managed by Alg. 6. As mentioned, these precedence relations are due to (1) resource constraints, i.e., the activities are allocated in the resources in a specific order in the generated enactment plan, and (2) SDeclare constraints related to precedence between activities. Algorithm 6 generates a map in which each scheduling activity is associated to a set of scheduling activities that are its direct predecessors (cf. Definition 32). For this, three maps are managed in this algorithm: (1) *directPredecessors*, which associates each scheduling activity to the set of its direct predecessors, (2) *indirectPredecessors*, which associates each scheduling activity to the set of its indirect predecessors (cf. Definition 33), and (3) *allPredecessors*, which associates each scheduling activity to the set of all its

direct and indirect predecessors. In Alg. 6, first, the precedences required due to the use of the same resource are included (lines 2-6). Secondly, the precedences required due to the high-level relations (i.e., SDeclare constraints) between the repeated activities which are stated in the model are included through the method `includePred` of each constraint (lines 7-8). Typically, unlike resource precedence relations, precedence relations due to SDeclare constraints cannot be easily obtained. To this end, each SDeclare template presents a method which is in charge of determining the precedence relations which are given between the scheduling activities related to the repeated activities which are involved in that SDeclare template. The mentioned method for some representative SDeclare templates is detailed in Algs. 7 and 8. Lastly, the indirect predecessors are removed from the map *directPredecessors* in order to avoid redundant connections, by taking into account that the sorted set *acts* is ordered by *st*, and hence, the scheduling activities are managed from minor to major *st* in the external loop (lines 9-15).

Algorithm 7: `includePred` method for the `Precedence` template with several source activities and one sink activity

input : Map<P&SAct,Set<P&SAct >> *pred*
output: Map<P&SAct,Set<P&SAct >> *pred*

1 Set<P&SAct> $meet \leftarrow \{a_1 \mid a \in this.sources, a_1.et \leq this.sink_1.st\}$;
2 P&SAct $sel \leftarrow argmin_{a \in meet}(a.et)$;
3 $pred(this.sink_1) \leftarrow pred(this.sink_1) \cup sel$;
4 **return** *pred*;

With respect to the `includePred` method, some representative templates are selected for illustration purposes (other templates can be described in a similar way). In Alg. 7, the template regarding the branched `Precedence` template with several source activities and one sink activity (i.e., it is modelled by a *ConstraintSources* object, cf. Figure 3.8) is shown. The location of a precedence template between several sources and one sink implies that the first execution of at least one of the sources must finished before the start of the first execution of the sink. In line 1, the set of scheduling activities which comply with the Precedence template (i.e, the first executions of the sources which end before the start of the first execution of the sink) are included in the set *meet*. At least one scheduling activity will be included in this set since the Precedence template is satisfied, however it may be possible to find more than one. In order to generate a BPMN model which is compatible with both the optimized enactment plan and the SDeclare specification, as is the purpose of the current approach, any scheduling activity of the set *meet* can be selected to be the predecessor of the sink in the BPMN model. One scheduling activity of the set *meet* is then selected to be the predecessor of the

sink. Specifically, the scheduling activity which presents more slack is selected (line 2) in order to construct a robust BPMN model. In line 3, the selected predecessor is included in the map, and is associated to the predecessors of the first execution of the sink. The fact that an activity B can start after another activity A has finished (ES, default option), is stated by including A in the set *pred* of B (line 3) of Alg. 8.

Algorithm 8: `includePred` method for the `AlternatePrecedence` Template with several source activities and one sink activity

input : Map<P&SAct,Set<P&SAct >> *pred*
output: Map<P&SAct,Set<P&SAct >> *pred*

1 Set<P&SAct> *meet* $\leftarrow \{a_1 \mid a \in this.sources, a_1.et \leq this.sink_1.st\}$;
2 P&SAct *sel* $\leftarrow argmin_{a \in meet}(a.et)$;
3 $pred(this.sink_1) \leftarrow pred(this.sink_1) \cup sel$;
4 **foreach** *i in 2..this.sink.nt* **do**
5 Set<P&SAct> *meet* $\leftarrow \{a_j \mid a \in this.sources, j \in 1..a.nt, this.sink_{i-1}.et \leq a_j.st \wedge a_j.et \leq this.sink_i.st\}$;
6 P&SAct *sel* $\leftarrow argmax_{a \in meet}((a.st - this.sink_{i-1}.et) + (this.sink_i.st - a.et))$;
7 $pred(sel) \leftarrow pred(sel) \cup this.sink_{i-1}$;
8 $pred(this.sink_i) \leftarrow pred(this.sink_i) \cup sel$;
9 **return** *pred*;

Allowing for Run-time Flexibility

The execution plans generated in Section 3.2 provide an optimal way for executing the source SDeclare model assuming certain estimated values and all decision to be goal-based. Even though these assumptions are valid for certain environments (e.g., certain web service settings) estimates might not always be accurate or some decisions might depend on run-time information. For this, the approach described in Sections 3.1 and 3.2 is extended in this section to allow decisions to be deferred at run-time, and to allow the BPMN model to be dynamically adapted during run-time.

Late-planning Activities

Executing a SDeclare model usually entails dealing with decisions related to (1) how many times one activity is being executed, and (2) the order of execution of the activities. This approach assumes that at least the decisions related to the order of execution of the activities are goal-based. However, non-goal-based decisions (e.g., user-based decisions) are considered, if needed, regarding the number

of executions of a particular activity. Related to these decisions, in turn, in Declare one activity can be executed arbitrarily often if not restricted by any constraint. However, there are some Declare templates which constrain the number of executions of the activities, resulting either in a specific value (e.g., A must be executed exactly twice), or in a range (e.g., A must be executed either once or twice). The number of times one activity should be executed can be stated by one specific constraint (e.g., Exactly(A,2)), or by the combination of several constraints (e.g., the combination of Exactly(A,2) together with ChainSuccession(A,B) implies that B should be executed exactly twice). To be able to deal with decisions related to the number of times certain activities are being executed which are not goal-based, this approach proposes to encapsulate these activities (together with the relations in which they are involved) in a complex declarative late-planning activity when specifying the SDeclare model, i.e., the use of hierarchical models is proposed. In declarative models the activities included in a complex activity should be such that they can be executed in isolation from the top-level process (Zugal et al., 2012).

Encapsulating decisions which are not goal-based in a fragment allows dealing with each sub-process (i.e., complex activity) as if it were a black box, and therefore, the current approach can be directly applied (even enabling multiple instance optimization). Therefore, when creating the optimized enactment plans from the SDeclare specification (cf. Section 3.2), each late-planning activity is treated as an atomic activity, and it is managed as a repeated activity (cf. Definition 26). In this way, when generating the BPMN model (cf. Section 34) the complex activities are then integrated into the BPMN model by substituting the BPMN activity related to the complex activity by the associated imperative fragment. For sake of clarity a description of how constraints, resources , and durations are managed is included:

- **Constraints**

  The BPMN fragment associated to a specific complex activity is generated as follows:

  1. Generating all possible combinations of declarative models in such a way that all different possibilities for *nt* (i.e., number of times) for each activity are covered. This is done by stating Exactly constraints for all the possible values for the number of executions for all the activities which belong to the complex activity. Specifically, for each activity A whose number of executions should be in a range [Min..Max], the generated models should cover all the possibilities (i.e., Exactly(A,nt), $\forall nt \in [Min..Max]$) in combination with all the possibilities for the other activities. Note that the maximum number of execution times

for each activity belonging to a complex activity needs to be established, otherwise, the possibilities are not finite.

2. For each declarative model which is generated, related optimized enactment plans are created (i.e., local optimization for each possible feasible declarative model is addressed) through the proposed constraint-based approach (cf. Section 3.2).
3. These optimized plans are then translated to BPMN fragments as stated previously in this Section.
4. These fragments are then linked by using existing merging algorithms (e.g., (Rosa et al., 2010)). Note that the resulting fragment will include XOR gateways when necessary.

When generating the different combinations of declarative models (i.e., step (1)) it is possible that some unfeasible combinations exist. In these cases, no related optimized enactment plan is generated, and therefore, the related BPMN fragment is not considered when merging (cf. Example 17).

Example 17. *Figure 3.9 shows an example of the complete process over a fragment which includes 5 BP activities (A, B, C, D and E) and 5 existence relations (i.e., all activities should be executed at most once) together with 5 binary relations (i.e., (1) ExChoice(A,C), implying that either A or C (but not both) must be executed, (2) ExChoice(B,D), implying that either B or D (but not both) must be executed, (3) Response(A,B), implying that after the execution of A, B should be eventually executed, (4) Precedence(C,D), implying that before the execution of D, C should be executed, and (5) Succession(D,E), implying that after the execution of D, E should be executed and before the execution of E, D should be executed). Given that declarative specification, there are 3 feasible scenarios, i.e., 3 possible ways to execute the specification ensuring that all constraints are satisfied:*

1. *A is executed once; C is not executed due to ExChoice(A,C); B is executed once after A due to the Response(A,B) constraint; D is not executed due to ExChoice(B,D), therefore also E cannot be executed due to Succession(D,E).*

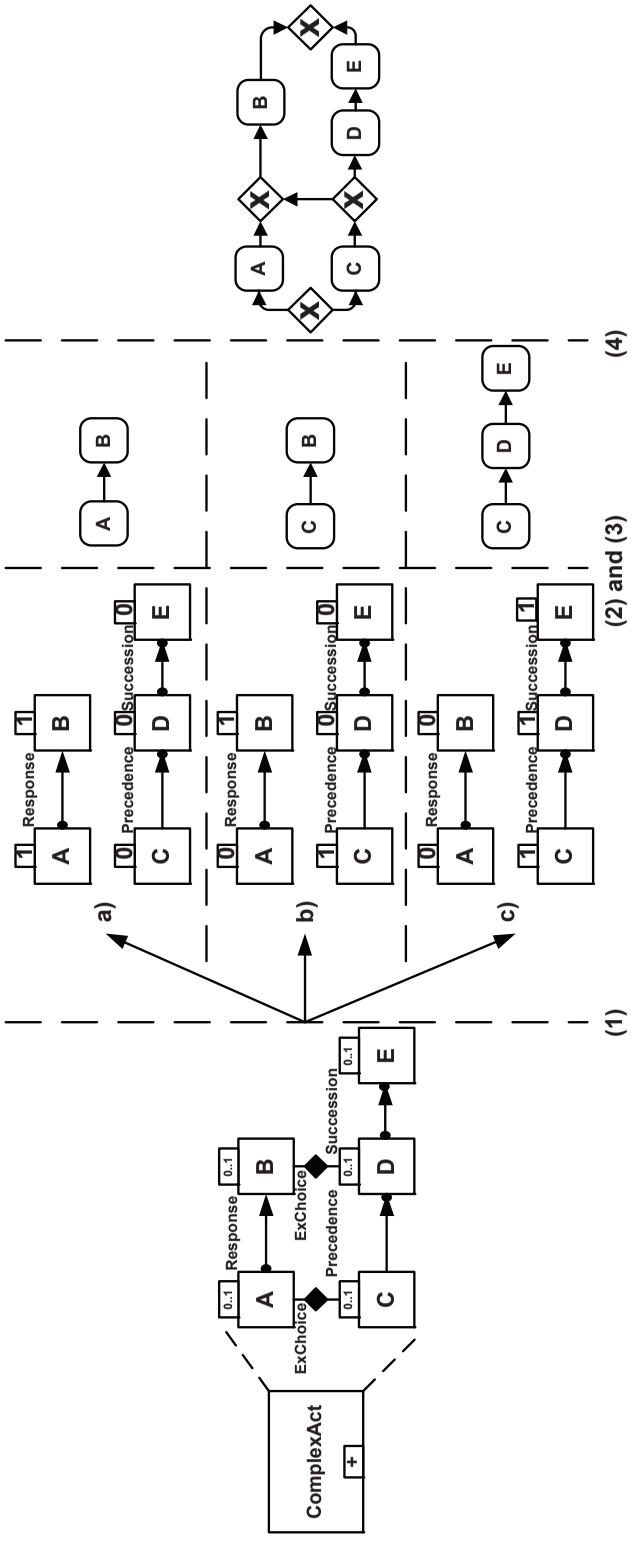

Figure 3.9: Generating BPMN fragments from declarative complex activities.

2. *C is executed once; A is not executed due to ExChoice(A,C); B is executed once; D is not executed due to ExChoice(B,D), therefore also E cannot be executed due to Succession(D,E). In the related optimized enactment plan, both options (B succeeding C or C succeeding B) are feasible. For this example, the option B succeeding C is considered more optimized than C succeeding B (note that for each feasible scenario only the most optimized plan is selected for the merging, as explained in the step (2) of the process).*
3. *C is executed once; A is not executed due to ExChoice(A,C)); D is executed once; B is not executed due to ExChoice(B,D). Since D is executed, E should be also executed due to Succession(D,E). In the related optimized enactment plan, C should precede D due to Precedence(C,D), and D should precede E due to Succession(D,E).*

In this example, some unfeasible combinations for nt exist. For example, the scenario in which A is executed once and D is executed once is unfeasible since 2 relations (i.e., Response(A,B) and Precedence(C,D)) are violated.

In Figure 3.9, the different BPMN fragments (related to the optimized enactment plans) which are obtained from the 3 feasible scenarios have been merged using the tool presented in (Rosa et al., 2010). For the sake of clarity, in Figure 3.9 information related to resources and durations of activities has been omitted.

Note that optimization is locally applied within each complex activity since for each declarative model which is generated (i.e., for each possibility) optimized enactment plans are generated.

- **Resources**

  For each complex activity, required resources need to be stated when including this activity in the SDeclare model. When all the activities which belong to the same complex activity require resources related to the same role, the complex activity will also require that role, and the proposed approach can be directly applied (cf. Figure 3.10(a), where all the activities require a resource of role R0). However, when the activities which belong to the same complex activity require resources related to different roles, some adjustments are required, e.g., encapsulating the declarative sub-process in a complex activity which requires as many resources as different roles are included in the sub-process (cf. Figure 3.10(b)), i.e., the constraint-based approach needs to be adapted to allow for activities which require multiple resources, resulting in a cumulative scheduling problem (Nuijten and Aarts, 1996a).

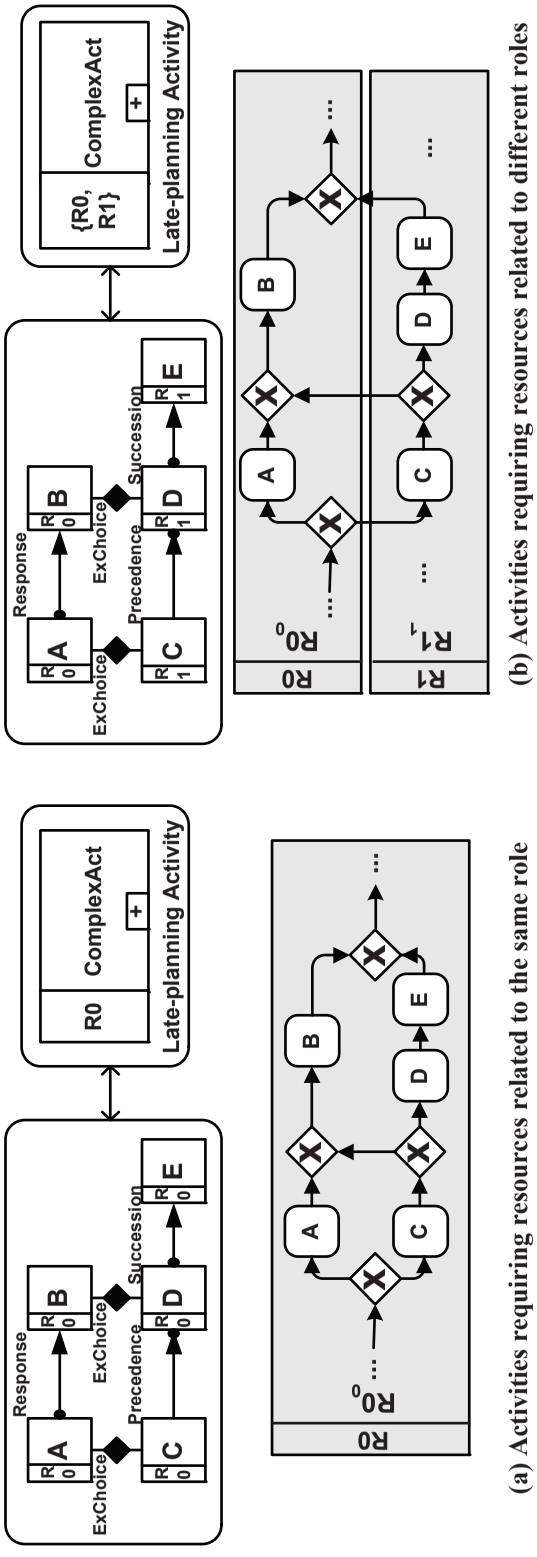

Figure 3.10: Complex activities: Dealing with resources.

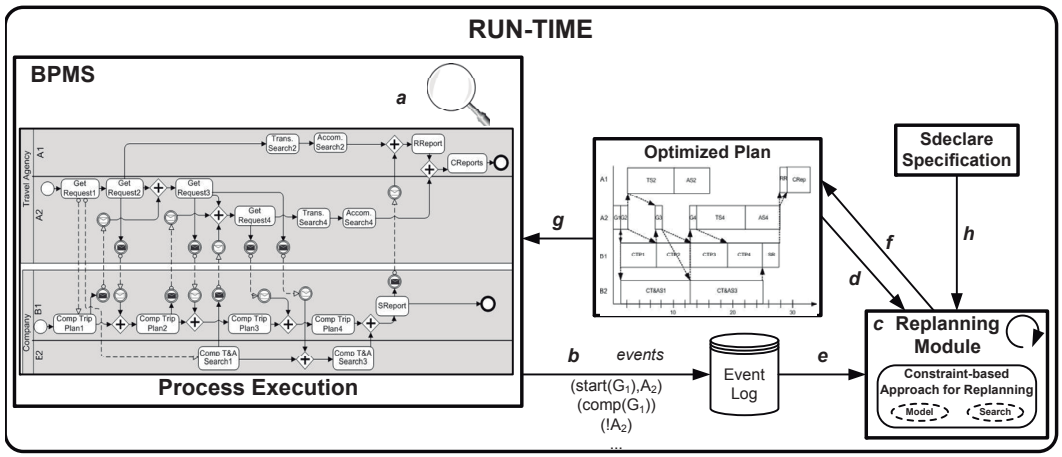

Figure 3.11: Flexible execution of BPMN models.

This extension can be easily achieved since most constraint-based systems provide a high-level constraint modelling specific to scheduling which includes an efficient management of shared resources for well-known scheduling problems, which is the case of the cumulative scheduling problem. When generating the BPMN model, each activity of the sub-process needs to be associated to the suitable lane (cf. Figure 3.10). Note that, in the proposed approach, the required resource is considered to be used throughout the duration of the activity.

- **Durations**

  Moreover, for each complex activity, estimated durations need to be stated when including this activity in the SDeclare model. The estimated durations of the complex activities can be calculated in different ways, e.g., as (1) the average duration of these complex activities in past process executions (i.e., by analyzing event logs), and hence, trying to optimize the resulting plan as much as possible although usually more replanning will be required, or (2) the maximum duration of these complex activities in past process executions, and hence, the plan is probably less optimized but less replanning will be required.

Replanning

Since estimates might not always be accurate and resource availabilities might unexpectedly change, the generated BPMN model is dynamically adapted during run-time by using replanning, and hence allowing for an increased flexibility (cf. Figure 3.11). As can be seen, as execution proceeds, the BP enactment and the

resource availabilities are monitored (a in Figure 3.11). All new events, i.e., activities get started/completed or resources become available/unavailable, are stored in an Event Log (b in Figure 3.11). Whenever events are updated the Replanning Module (c in Figure 3.11) analyzes the optimized plan (d in Figure 3.11) as well as the events. In particular, it checks if the current execution matches with the optimized enactment plan or whether updates of both the enactment plan (f in Figure 3.11) and the BPMN model (g in Figure 3.11) are required. In general, updates can become necessary due to (1) deviations, i.e., estimates are incorrect (e.g., when activity executions take longer/shorter than estimated), or (2) resource availabilities change (e.g., resources become unavailable). Note that not every deviation requires replanning due to the slack of some activities in the enactment plan. If plan updates are required, the Replanning Module needs to access the SDeclare specification of the process (h in Figure 3.11) to generate a new optimized plan which considers the actual partial execution of the process by using the proposed constraint-based approach (cf. Section 3.2). The generated optimized plan is, in turn, translated to an optimized BPMN model which is used for updating the current BPMN model in a way that the part which has been already executed remains unchanged, and the part which remains to be executed is replaced. Despite the NP-complexity of the considered problems, in general, 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.

Note that changing a deployed BPMN model and migrating running instances to a new schema can be quite challenging since respective changes must not violate process model correctness and proper instance execution (Reichert and Weber, 2012). However, in the current approach, the proposed model adaptation and instance migration can be handled properly as detailed in the following.

On the one hand, in process model evolution it is necessary to check that the new model is (1) correct, i.e., it meets the structural properties required by the process modelling language used, and (2) sound, i.e., it obeys proper completion and absence of dead activities (Reichert and Weber, 2012). In the current approach, the generated BPMN model is correct since the automated generation guarantees that the new model meets the structural properties required by BPMN. Moreover, it is sound since the model is automatically generated from a feasible enactment plan which meets all the constraints imposed by the declarative specification and reaches the specified goal. Since the generation of the new models is not manual but completely automated, no errors can be unintentionally introduced.

On the other hand, once a new correct and sound model is deployed, the BPMS must properly deal with corresponding process instances, i.e., process instances which were started and partially executed on the previous model, but have not yet been completed (Reichert and Weber, 2012). In this way, in addition to struc-

tural properties, the BPMS needs to consider the state of a process instance when adapting its process model (Reichert and Weber, 2012), i.e., depending on the current state of a process instance, certain changes should be allowed while others must be prohibited (e.g., it must not be possible to change the past of a process instance). Specifically, the running process instance should be state compliant with the new process model. A process instance is state compliant with an updated process model (i.e., can therefore be migrated to it) if the execution trace of the instance is producible on the new model as well. In the case of this Thesis, there is only one running instance (which comprises the execution of all instances which were planned within a specific timeframe) which has to be migrated to the new model version. This is not problematic in the current approach since the new model is generated through replanning from the partial execution trace of this instance. Therefore, this trace will be always producible on the new model, i.e., everything which has been done before can be done in the new model. However, for migrating this instance to the new process model version, activity states might have to be adapted to enable proper continuation of instance execution afterwards. As an example of instance state adaptation, it might become necessary to immediately enable or disable certain activities before continuing with the execution of the process instance (Reichert and Weber, 2012). Using selected commercially available state-of-the art BPMSs (e.g., AristaFlow BPM Suite (AristaFlow, 2009)) respective changes can be accomplished.

3.4 Related Work

This chapter significantly improves and extends the Declare language by considering multi-objective optimization, choice (Pesic, 2008), temporal (Montali, 2009; Westergaard and Maggi, 2012) and data constraints (Montali, 2009; Montali et al., 2013), and alternative resources. Hence, more realistic problems and more expressive specifications can be managed. In fact, SDeclare is based on the time extension defined in (Montali, 2009) where it is possible to define time lags over the different Declare constraints. The same time-aware extension is considered in (Westergaard and Maggi, 2012) where, additionally, a deep reasoning based on a finite automaton is performed to warn the users to avoid wrong states. Furthermore, a data-aware extension has been recently proposed in (Montali et al., 2013). Such extension is considered in the current approach. Nevertheless, unlike the current approach, (Montali et al., 2013) is based on Event Calculus and it is focused on monitoring and operational support.

This Thesis is not aware of any other approaches for generating set of enactment plans from declarative specifications. However, there exist some further proposals which could be extended in such direction (Pesic, 2008; Montali, 2009;

Krogt et al., 2010; Lu et al., 2009; Rychkova et al., 2008; Hummer et al., 2013). Specifically, (Pesic, 2008) proposes the generation of a non-deterministic finite state automaton from constraint-based specifications which represents exactly all traces that satisfy the constraints. However, the big disadvantage following such an approach would be that the process of generating the automaton from the declarative specifications is exponential with respect to the size of the formula (Montali et al., 2010), and, unlike the proposed approach, no heuristic is used. Additionally, CLIMB (Montali, 2009) could be used to generate quality traces from declarative specifications, and calculate its values for different objective functions. Then, the best traces could be selected. Unlike the proposed approach, (Montali, 2009) does neither consider optimality nor resource availabilities. Therefore, these would only cover the planning part of the current proposal, but not the scheduling aspects. In a related way, the work (Krogt et al., 2010) plans and schedules tasks considering resources and the optimization of one objective function through an integer constraint-based specification. Although (Krogt et al., 2010) presents a similar constraint-based approach, it misses dealing with multi-objective optimization, and does not support high level constraints. Moreover, in (Lu et al., 2009), a constraint formalization is proposed to generate variations of an ad-hoc BPMN sub-processes. In a similar way, (Rychkova et al., 2008) proposes the specification of processes based on a first-order logic language and translates them to an imperative language. In turn, related to BP, (Hummer et al., 2013) provides a model-driven approach which produces an imperative process specification from a declarative specification. Unlike the current approach, (Lu et al., 2009; Rychkova et al., 2008; Hummer et al., 2013) do not consider the optimization of any objective function.

Several filtering rules for specialized scheduling constraints have been developed. Specifically, (Bartak and Cepek, 2010; Laborie et al., 2009) model scheduling problems which include alternative and optional tasks respectively, together with their filtering rules. The proposed model and propagation for the optional activities in the current work are very similar to the proposal presented in (Laborie et al., 2009). However, unlike (Bartak and Cepek, 2010; Laborie et al., 2009), to efficiently manage SDeclare constraints complex and innovative filtering rules are developed which are related to the alternating executions of repeated activities together with the variable number of times which these activities are executed.

Chapter 4

Guiding the Optimized Execution of Constraint-based BP Models subject to Uncertainty through Questionnaires

4.1 SDeclare 2.0: Extending SDeclare 1.0 by Including Stochastic Estimates

As stated in Section 3.1, to specify the processes in a declarative way, Declare (Pesic, 2008) is used as basis. In this chapter, the second version of the SDeclare language is proposed by considering stochastic estimates.

Stochastic Estimates

As mentioned, to allow the specification of certain input uncertainty in the declarative BP models which are designed, Declare is extended by including the **stochastic attributes** for certain parts of the model (i.e., S-Activity attributes, data and temporal constraints, and resource availability). Estimates can be obtained by interviewing business experts or by analysing past process executions. Moreover, both approaches can be combined to get more reliable estimates.

Since estimating values can be quite challenging (Souki, 2011), SDeclare allows specifying any discrete value of the model in a stochastic way by using probability mass functions (*PMFs* in the following) which are functions that give the probability of a variable taking a certain value. These PMFs can be associated to any input data (cf. *Data* in Definition 25) of the SDeclare model. Thus, the Data property of a SDeclare model consist of tuples $<dName,dValue/dPMF>$,

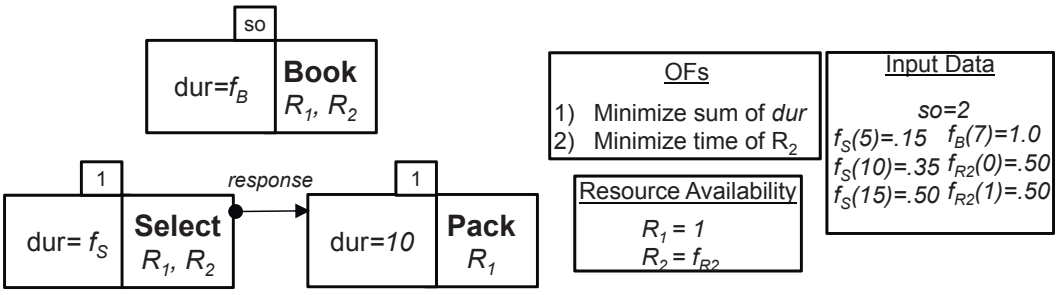

Figure 4.1: Example of SDeclare process model

meaning that the data *dName* can be associated to a fixed value (*dValue*) or to a PMF (*dPMF*) (cf. Example 18).

Example 18. *In Figure 4.1, a SDeclare process model related to how to prepare a holiday is depicted. To prepare a holiday three activities need to be performed, select the cloths, pack them, and book the flights between each stopover. The objective is to minimize the planning time and, additionally, to minimize the use of a second resource. Therefore, SActs* = $\{Book, Select, Pack\}$, *OFs*= {minimize sum of dur, minimize time of R_2}, *AvRes* = $\{<R_1,1>, <R_2, f_{R2}>\}$ *(i.e., the number of available resources of* R_2 *is defined by the PMF* f_{R2}*), Data* $=<so, 2>$ *and* C_{BP}=$\{$*Response(Select ,Pack), Exactly(1, Select), Exactly(1, Pack), Exactly(so, Book)*$\}$ *(i.e., the number of repetitions of Book is defined by the input data so). In addition,* $\{Book, Select, Pack\}$ *are three S-Activities (cf. Definition 24) where Book=*$<Book, \{R_1, R_2\}, \{<dur, f_B>\} >$, *Select=*$<Select, \{R_1, R_2\}, \{<dur, f_s>\} >$ *(i.e., its durations are defined by the PMFs* f_B *and* f_S*), and Pack=*$<Pack, \{R_1\}, \{<dur, 10>\} >$. *Note that there is uncertainty related to the duration of Select, i.e., it may last 5, 10 or 15 units of time and 15 is the most probable value.*

Using PMFs, the estimates reflect the business reality better (AbouRizk et al., 1994). There are extensive studies focused on patterns of PMFs that represent the uncertainty best (AbouRizk et al., 1994; Fente et al., 2000; Maio et al., 2000; Back et al., 2000) that are not discussed here since it is out of the focus of this Thesis Dissertation.

4.2 Generating Configurable BP Models

In this section, the generation of a configurable BP model is explained. This includes: (1) the sampling of the stochastic properties of the SDeclare model to obtain a set of non-stochastic models (cf. Section 4.2.1), (2) the generation of optimized enactment plans for such non-stochastic models (cf. Section 4.2.2), (3) the definition of two properties to measure how well the input uncertainty is

managed by the plans (i.e., flexibility and robustness, cf. Section 4.2.3), (4) the filtering of non-desirable enactment plans (cf. Section 4.2.4), and (5) the merging of the resulting plans in a configurable BP model (cf. Section 4.2.5).

4.2.1 Sampling the SDeclare Model

As stated in Section 4.1, stochastic properties can be included in the SDeclare model (e.g., S-Activity attributes, data properties or resource availabilities). These properties represent the input uncertainty that is considered in the scenario and it is used to evaluate the flexibility of the configurable BP model and the robustness of the optimized enactment plans included in the model, as explained later.

For managing the uncertainty of the SDeclare model when generating the related optimized enactment plans, the different stochastic properties are sampled (cf. Definition 35) by considering their associated PMFs.

Definition 35. *Let* $SDM = (SActs, Data, C_{BP}, AvRes, OFs)$ *be a SDecare model with n stochastic properties* $prop_1, .., prop_n$*. Then: a* ***sample*** *is a set of n tuples* $< prop_i, val_i >, i = 1..n$ *which indicates the fixed value* val_i *that the property* $prop_i$ *takes in such sample. The value* val_i *is randomly selected considering the PMF related to* $prop_i$ *(i.e.,* f_{prop_i}*).*

Each sample is used to create a non-stochastic model (cf. Definition 36) from a SDeclare model by assigning a fixed value to each stochastic property (cf. Example 19). In the proposed approach, multiple samples are generated in order to obtain a representative set of non-stochastic models. Each non-stochastic model is, in turn, transformed to a MO-COP.

Definition 36. *A* ***non-stochastic SDeclare model*** *is a SDeclare model in which all properties are defined by fixed values.*

Example 19. *Regarding the SDeclare model of Figure 4.1, a possible sample could be:* $\{< fS, 10 >, < fB, 7 >, < fR2, 0 >\}$*. Applying the sample to the SDeclare model, the non-stochastic SDeclare model of Figure 4.2 is obtained.*

4.2.2 Generating Multi-objective Optimized Plans

The SDeclare model $SDM = (SActs, Data, C_{BP}, AvRes, OFs)$ is initially modified by the samples which are considered in such a way that one non-stochastic model is generated for each sample. For generating the multi-objective optimized enactment plans, the approach stated in Chapter 3 is applied to each non-stochastic model. Then a set of BP enactment plans are obtained.

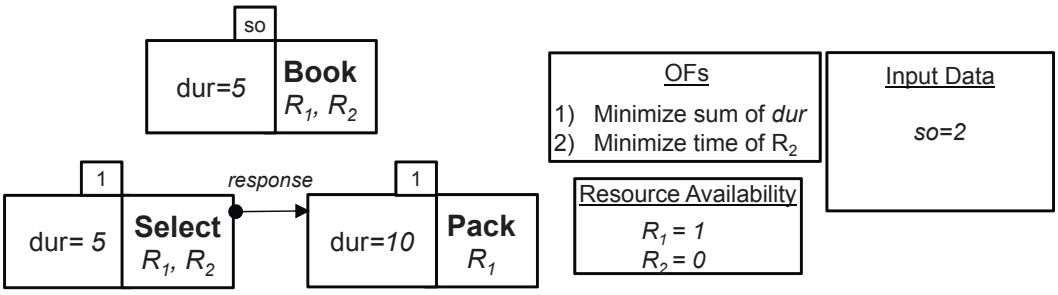

Figure 4.2: A non-stochastic SDeclare model resulting from applying the sample of Example 7 over the SDeclare model of Figure 4.1.

4.2.3 Quantifying the Flexibility and the Robustness

In this section definitions for both flexibility and robustness are proposed in order to measure how the generated models deal with the uncertainty. Such definitions are based on related literature, which is introduced in Section 2.4. In the BP field, flexibility and robustness can be treated as quantitative attributes related to a specific process model. In the context of this Thesis, robustness and flexibility are quantified over enactment plans (cf. Definition 1) and configurable BP models (cf. Definition 21) respectively.

As mentioned in Section 2.3.2, a configurable BP model includes different enactment plans sice there exists a one-to-one relation between graphs and enactment plans. Each enactment plan which is included in such models has its own level of robustness against a specific variable which presents uncertainty (cf. Definition 37). This uncertainty is defined through the related PMF which is included in the SDeclare model (cf. Definition 25).

Definition 37. *Let P_i be an enactment plan (cf. Definition 1); let v be a variable related to some attribute of P_i and which is defined in the domain $D(v)$ with a PMF $f_v : D(v) \rightarrow [0..1]$, $\sum_{x \in D(v)} f_v(x) = 1$; and let $W(P_i, v)$ be the set of values of v which P_i withstands, i.e., P_i tolerates scenarios where v takes any value in $W(P_i, v)$ without changing its performance (e.g., without changing its objective function values, cf. Definition 20). Then: the* ***robustness of*** *P_i* ***against*** *v, $Rob(P_i, v)$, is the probability of the variable v taking a value that P_i withstands. When v is a discrete variable, then $Rob(P_i, v) = \sum_{x \in W(P_i,v)} f_v(x)$. When v is a continuous variable, $W(P_i, v)$ is considered as the non-overlapped ranges of values of v (i.e., $[[r1_{inf}, r1_{sup}], [r2_{inf}, r2_{sup}]...]$) which P_i withstands; then $Rob(P_i, v) = \sum_{rx \in W(P_i,v)} (\int_{rx_{inf}}^{rx_{sup}} f_v(x)dx)$.*

In this way, the *robustness against a variable* is applied over single alternatives (i.e., single enactment plans) of a configurable BP model. By contrast the term

flexibility (cf. Definition 38) is applied over configurable BP models (cf. Example 20).

Definition 38. *Let P be a configurable BP model which contains different enactment plans $P_i \in P$; and let v be a variable related to some attribute of P and which is defined in the domain $D(v)$ with a PMF $f_v : D(v) \rightarrow [0..1]$, $\sum_{x \in D(v)} f_v(x) = 1$. Then: the **flexibility of** P **against** v, $Flex(P,v)$, is the probability of the variable v taking a value that P withstands by adapting its workflow to any of its alternatives P_i. When v is a discrete variable, then $Flex(P,v) = \sum_{x \in \bigcup_{P_i \in P} W(P_i,v)} f_v(x)$. When v is a continuous variable, $W(P_i,v)$ is considered as the non-overlapped ranges of values of v (i.e., $[[r1_{inf}, r1_{sup}], [r2_{inf}, r2_{sup}]...]$) which P_i withstands; then $Flex(P,v) = \sum_{rx \in \bigcup_{P_i \in P} W(P_i,v)} (\int_{rx_{inf}}^{rx_{sup}} f_v(x)dx)$.*

When the variable v follows a flat distribution, the robustness and the flexibility can be expressed as $Rob(P_i,v) = |W(P_i,v)|/|D(v)|$ and $Flex(P,v) = |\bigcup_{P_i \in P} W(P_i,v)|/|D(v)|$ respectively.

Example 20. *Consider the two enactment plans depicted in Figure 4.3 (a), and the two probability mass functions shown in Figure 4.3 (b) (i.e., f_{R2} which is related to the number of available resources with role R2 and f_S which is related to the duration of the activity Select, i.e., S). Then, some measures can be calculated (cf. Figure 4.3 (c)). The robustness of the enactment plan P_1 against S, $Rob(P_1,S)$, is equal to 0.15 since P1 only withstands that activity Select takes 5 units of time. However, the robustness of P1 against R2, $Rob(P1,R2)$, is equal to 1 since P1 is valid for any availability of R2 (note that R2 is not used in P1). In a similar way, the robustness against these 2 variables is calculated for plan P2. Considering the last column of the table Robustness, it can be concluded that the enactment plan P2 manages the uncertainty better that P1. In a related way, once a configurable BP model is created by merging these two enactment plans, then the flexibility of such model can be calculated as stated in Definition 38. Therefore, $Flex(P,R2)$ is equal to 1, since P includes plans which can withstand any value of R2, and $Flex(P,S)$ is equal to 0.5, since the value 15 for S is not withstood by any plan of P. Considering both variables together, $Flex(P,R2 \wedge S) = 0.325$ which means that the 32.5% of the input uncertainty is properly managed by the configurable BP model.*

4.2.4 Selecting the Relevant Plans

In order to select the relevant plans from the set of optimized enactment plans (denoted by *PS* from now on) a two-steps algorithm is proposed:

1. Considering that the uncertainty of the scenario is specified over the stochastic variables (i.e., v_p) associated to some properties of the SDeclare model,

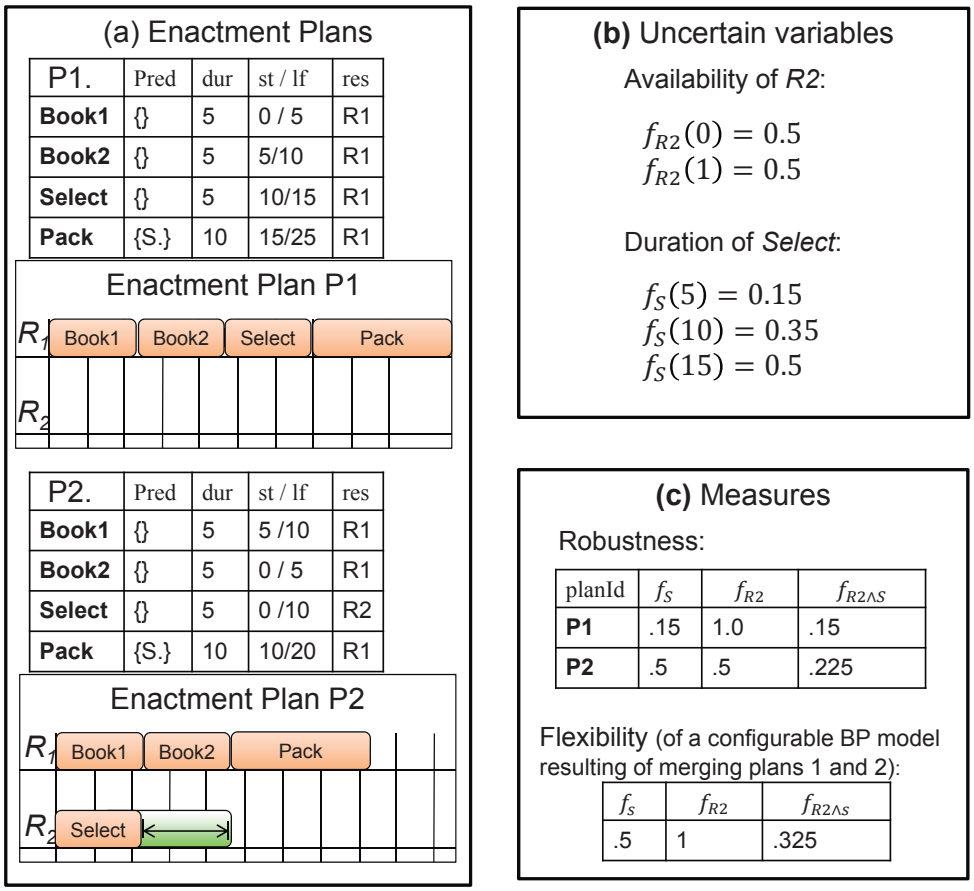

Figure 4.3: For two different enactment plans (a) generated from the SDeclare model of Figure 4.2, and considering two uncertain variables (b), the robustness of each plan and the flexibility of the related configurable BP model are calculated (c).

a set of properties are calculated for each enactment plan $P_i \in PS$ (cf. Example 21):

- Withstood values for each uncertain variable: For each v_p, the range of withstood values are calculated (i.e., $W(P_i, v_p)$). Note that calculating the withstood ranges of the S-Activity attributes or of the availability of resources might be trivial. However, when the uncertainty is specified over data properties which affect a constraint, calculating the withstood ranges may require more elaborated calculus.
- Robustness for each uncertain variable: The robustness against each uncertain variable v_p (i.e., $Rob(P_i, v_p)$) is calculated as stated in Definition 37.

Table 4.1: Variables S and R_2 which are defined by the PMFs f_S and f_{R_2} respectively.

S	f_S	R_2	f_{R2}
5	0.15	0	0.5
10	0.35	1	0.5
15	0.5		

Table 4.2: Properties which are calculated for the set of enactment plans of Figure 4.3.

Enactment Plans			Measures				
$Plan_{id}$	*sum dur*	*time* R_2	$W(P_i,S)$	$W(P_i,R_2)$	$Rob(P_i,S)$	$Rob(P_i,R_2)$	$\overline{Rob}(P_i)$
P1	25	0	[5]	[0, 1]	0.15	1.0	0.575
P2	20	5	[5, 10]	[1]	0.50	0.5	0.5

- Average robustness: Finally, the general robustness of a plan (i.e., $Rob(P_i)$) is calculated as the mean of the individual robustness of each stochastic variable.

Example 21. *Table 4.2 shows a set of 2 optimized enactment plans (cf. column $Plan_id$) generated from the SDeclare model shown in Figure 4.1 with two uncertain variables f_s and R_2 (cf. Table 4.1 and its associated objective function values (cf. columns sumdur and time R_2). Moreover, regarding the domains that each plan withstands against v_1 and v_2 (cf. columns $W(P_i,v_1)$ and $W(P_i,v_2)$ respectively), the value of the robustness against these variables can be calculated as stated in Definition 37. The values of the robustness are depicted on columns $Rob(P_i,v_1)$ and $Rob(P_i,v_2)$ respectively. Furthermore, the value of the general robustness (cf. column $Rob(P_i)$) is calculated as the mean of $Rob(P_i,v_1)$ and $Rob(P_i,v_2)$.*

2. In this step, the relevant plans are selected. For this, three different policies can be considered:

   (a) All plans are kept: No plan is removed. In this case, the non-desirable variability is not reduced.

   (b) The plans which present the highest robustness are kept: The enactment plans are ranked by its average robustness. Then, a percentage of plans which present the lowest robustness are removed (cf. Example 22). The goal of this policy consists of creating a configurable BP

model composed by the most robust plans to the detriment of the flexibility, i.e., this policy is not intended to cover the input uncertainty as much as possible.

Example 22. *In Table 4.2 the plan p2 presents an average robustness which is significantly lower than the robustness of the other plans, and then it would be removed if this policy is followed.*

(c) The plans which provide for the highest flexibility are kept: The minimum set of plans which covers the maximum input uncertainty is selected (i.e., the minimum set which maximizes the union of the withstood domains, cf. Example 23). The goal of this policy consists of creating a configurable BP model which provides for the highest flexibility, i.e., which embraces plans which cover as much uncertainty as possible typically to the detriment of the robustness of these plans.

Example 23. *When following this policy, in Table 4.2 the plans P1 and P2 would be selected since they are not totally overlapped.*

In this way, this second step removes some enactment plans regardless of their objective function values. Therefore, *good* plans (i.e., *optimized* plans) which were calculated in Alg. 1 are removed here, and then, only those plans which are *relevant* (i.e., the plans which are selected according to a policy) remain. This way, only the plans which are both good and relevant are kept.

The proposed approach could be easily adapted to consider the robustness as an additional objective function when generating the set of optimized enactment plans. However, in that scenario non-optimal solutions would be included since a new dimension would be considered in Alg. 1, i.e., the robustness.

4.2.5 Merging the Relevant Plans into a Configurable BP Model

As stated in Section 2.3.2, an adaptation of the Process Merger tool (Rosa et al., 2012) is used to create the configurable BP model out of the selected plans. The enactment plans to be merged are identified by a label, i.e., *pid* attribute (cf. Definition 1, column *Plan_id* in Table 4.2, Figure 4.4 (a)). The generated configurable BP model has special nodes called configurable nodes which represent the variation points of the model (cf. Figure 4.4 (b)). In addition, each arc of the configurable BP model has a reference to the labels of the plans to which the arc belongs. The variant to be executed is selected from the configurable BP model before the run-time phase regarding (1) the actual values of the uncertain variables of the scenario, (2) the robustness of the plans which withstand such actual values,

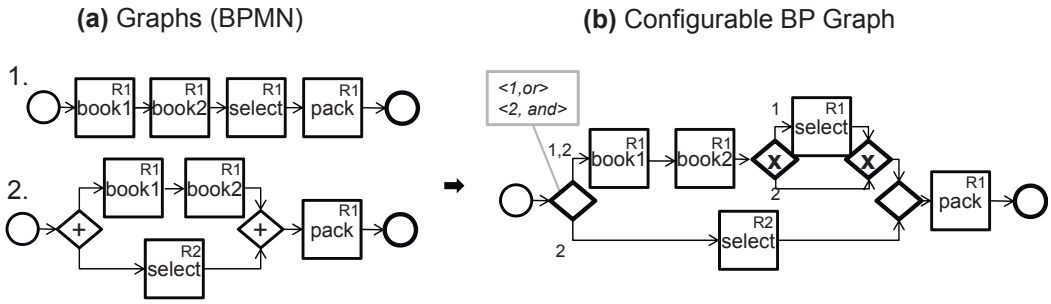

Figure 4.4: Two different BPMN Graphs (a) related to the enactment plans of Figure 4.3 merged into a configurable BP model (b).

and (3) the values of the objective functions. Just in case the flexibility which is obtained becomes insufficient (i.e., none of the enactment plans which are in the configurable BP model withstands the actual values of the uncertain variables), replanning becomes necessary and new optimized BP enactment plans will be generated by considering the actual values of the uncertain variables instead of the PMFs and then a new configurable BP model will be created.

4.3 Run-time Individualization of Configurable BP Model

In this section, a method for automatically generating questionnaires from a declarative model and its usage for supporting the user during the execution of such model is described (cf. Figure 4.2). Such a method uses a configurable BP model as starting point with can be generated as detailed in Section 4.2. Then, the BP execution starts and advances until a configurable node (cf. Definition 21) is found in the configurable BP model (cf. Section 4.3.1, Figure 4.5 (1)). Thereafter, a decision tree related to such configurable node is created (cf. Section 4.3.2, Figure 4.5 (2)) as an intermediate step for generating the questionnaire associated to this configurable node (cf. Section 4.3.3, Figure 4.5 (3)). Whenever the user answers a questionnaire (i.e., a decision is taken, cf. Section 4.3.4, Figure 4.5 (4)), the variants of the configurable BP model are narrowed down based on the answers given. This method is iteratively applied from step 1 to step 4 until no more individualization is needed (i.e., until only one single variant remains in the configurable BP model).

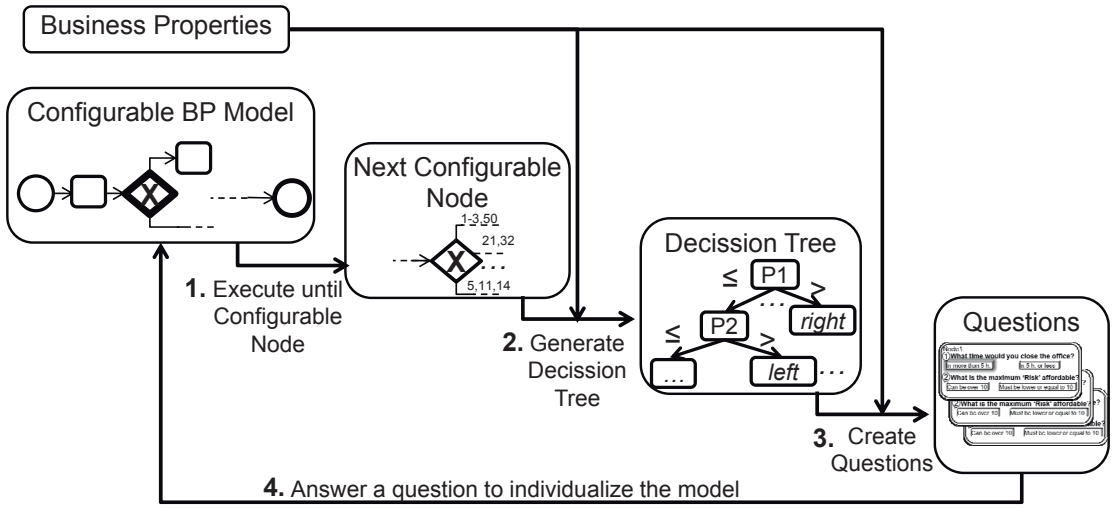

Figure 4.5: Automatic generation of questionnaires for Individualizing a configurable BP model.

4.3.1 Execution the Configurable BP Model

As stated in Definition 21, all variants which are included in the configurable BP model are labeled (cf. Example 24).

Example 24. *The running example of Figure 4.6(a) comprises four BP models, each one labeled with an integer. Furthermore, a group of properties for each BP model is provided (cf. Figure 4.6 (b) where time (T), benefit (B) and risk (R) properties are provided for each model). Such properties are related to the business language, e.g., T is related to the opening hours of the business. The configurable BP model associated with the BP models which are depicted in Figure 4.6 (a) is shown in Figure 4.6 (c). In this model, 4 different configurable nodes are depicted with a bold diamond. In the first configurable node, labeled as 1, two alternatives are possible. The lower branch comprises BP Model 4 (i.e., where activity A is not executed), and the upper branch comprises BP Models 1 to 3 (where activity A is executed).*

The configurable BP model can be executed from the beginning until a configurable node appears, i.e., until a decision must be taken (cf. Figure 4.5 (1)).

Note that the selection of a valid variant is guaranteed since this approach building upon previous work which generates valid variants. Merging them preserves these variants and the same happens with the decision trees.

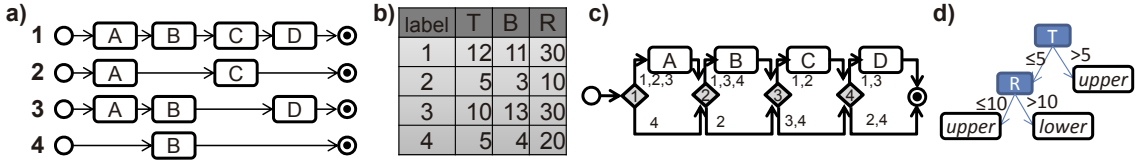

Figure 4.6: a) 4 different BP models. (b) Properties of each BP model. (c) configurable BP model related to the BP models of (a). (d) Classification tree for node 1.

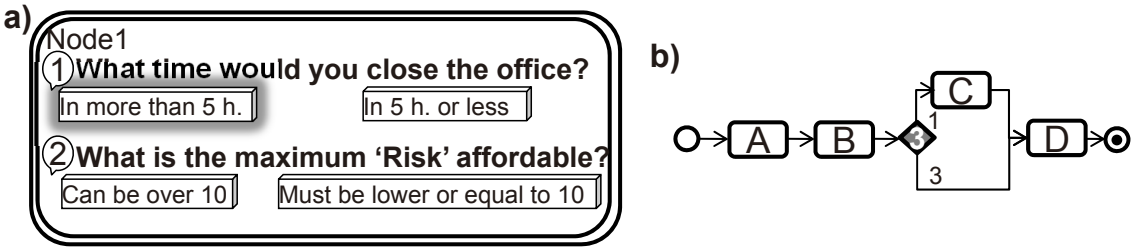

Figure 4.7: (a) Questionnaire for Node 1. (b) The resulting configurable model after removing Variants 2 and 4.

4.3.2 Generating Decision Trees

When a configurable node is encountered the current approach applies a method for generating a prediction system (i.e., a model that predicts the value of a target variable based on several input variables) (Breiman, 1984) for predicting which outgoing branch corresponds to a given assignment of property values. Specifically, for each configurable node, a classification tree is created (cf. Figure 4.5 (2)) using the property values of the variants as input variables and the outgoing branches as target variables (cf. Example 25).

Example 25. *Figure 4.6(d) shows the classification tree which comes of using the CART algorithm (Breiman, 1984) when using the table of Figure 4.6(b) as input variables and the strings lower and upper as target variables. As can be seen, in the resulting classification tree, the variants for which $T > 5$ correspond to the upper branch. In contrast, the variants for which $T \leq 5$ correspond to the upper branch if $R \leq 10$, or to the lower branch otherwise.*

4.3.3 Creating Questions

A set of questions is then created for each decision tree (cf. Figure 4.5 (3)). To create such questions according to the business language, a set of well-defined business properties must be provided. This way, one question is automatically generated for each intermediate node of the tree. The possible answers for such

question are the different labels which are written on the outgoing branches of this node. The text of the questions is automatically generated from the information of the provided business properties (cf. Example 26). As stated, these questionnaires are in charge of narrowing down the variants of the configurable BP model.

Example 26. *A simple questionnaire related to the decision tree of Figure 4.6(d) is shown in Figure 4.7(a). Since this decision tree has two intermediate nodes (i.e., T and R), two questions are created. Moreover, since each node has two branches, each question has two options. Initially, only the question related to T is enabled. Considering that the well-defined business properties stated that T is related to the* closing time *of the office, the generated question would look like* What time would you close the office?.[1] *The second question has to be answered only if the user selects the second option of the first question (i.e.,* In 5hrs. or less*) which is related to the branch* $T \leq 5$ *of the decision tree.*

4.3.4 Incremental Configuration

Whenever a questionnaire is answered, the configurable BP model is narrowed down by removing the variants that do not belong to the edge selected in this configuration step. Thereafter, the proposed method continues at Step 1 (cf. Figure 4.5) considering the narrowed configurable BP model and continuing the execution from the last executed activity.

Such method is repeated until only one variant remains in the configurable BP model, i.e., the configuration has finished (cf. Example 27).

Example 27. *Supposing that the user selects the first answer of the first question of the questionnaire of Figure 4.7(a) (i.e.,* In more than 5hrs.*), Variants 2 and 4 are removed from the configurable BP model since they have a time property "*≤ 5*". This results in the configurable BP model of Figure 4.7(b). Note that the second and forth configurable node of Figure 4.6(c) are not depicted in Figure 4.7(b) since Variants 1 and 3 share the same outgoing branches for these nodes, i.e., the upper branch. However, the third configurable node requires selecting one of the two branches, and hence, a new questionnaire is generated.*

4.4 Related Work

The Declare language (Pesic, 2008) has previously extended in Chapter 3. In this chapter, as a major contribution of SDeclare 2.0 regarding existing proposals

[1] Note that, the semantic of the generated questions highly depends on the information provided for the business properties. Such information can be used to make the questions more user-friendly. No depth details are given since it is out of the scope of this Thesis.

(Westergaard and Maggi, 2012; Montali et al., 2013; Montali, 2009), it allows specifying the input uncertainty of real scenarios by using stochastic values.

In addition, none of the approaches which can be used to generate imperative models from declarative specifications (Pesic, 2008; Montali, 2009; Krogt et al., 2010; Lu et al., 2009; Rychkova et al., 2008; Hummer et al., 2013) considers the uncertainty of the scenario through stochastic attributes.

As mentioned, several approaches exist for dealing with flexibility issues (Cicerone et al., 2012; Aissi and Roy, 2010), even in the context of BP (Reichert and Weber, 2012; Schonenberg et al., 2008a). However, in this chapter as a novel contribution, it proposes quantitative definitions for both robustness and flexibility which allow an analyst to measure how the uncertainty of a real scenario is supported by an enactment plan and by a configurable BP model respectively.

In literature, different approaches deal with the variability of BPs (Schnieders and Puhlmann, 2006; Hallerbach et al., 2010; Rosa et al., 2011; Kumar and Yao, 2012; Rosemann and van der Aalst, 2007; Schunselaar et al., 2012). In PESOA (Schnieders and Puhlmann, 2006) and C-EPC (Rosemann and van der Aalst, 2007; Rosa et al., 2011) configurable BP models are used as basis. Variation points are either defined by a set of annotations of activities (Schnieders and Puhlmann, 2006) or by including configurable BP nodes (Rosemann and van der Aalst, 2007; Rosa et al., 2011). In turn, the work (Rosa et al., 2011) improves the C-EPC language by incorporating both resource and data notation in the C-EPC models. Unlike C-EPC and PESOA, Provop (Hallerbach et al., 2010) and RULE (Kumar and Yao, 2012) consider a base process model which is configured using a set of predefined change operations (Hallerbach et al., 2010) or applying some business rules (Kumar and Yao, 2012). Furthermore, there are other approaches which mix declarative specifications with configurable BP models, e.g., ConfDeclare (Schunselaar et al., 2012) presents a declarative language in which activities can be hidden and constraints can be omitted. Regarding how configurable BP models are created, essentially, there are two ways: manually and automatically. On the one hand, the manual creation of configurable BP models can be carried out from scratch by manually specifying the variation points (Schnieders and Puhlmann, 2006; Hallerbach et al., 2010; Kumar and Yao, 2012; Rosemann and van der Aalst, 2007; der Aalst et al., 2006; Gottschalk et al., 2008). The main problems of the manual creation is that it is typically a very time consuming task and requires deep skills on the modelling language. On the other hand, an automatic method has been proposed to generate a configurable BP model, in C-EPC language, from a set of BP models by analyzing the similarities of the source BP models and including variation points where they differ (Rosa et al., 2010, 2012). The main problem of the automatic creation of configurable BP models is that it requires a family of BP models independently specified. Since this chapter pro-

poses to automatically create configurable BP models, the current approach builds upon these techniques.

Chapter 5

Empirical Evaluation

5.1 Introduction

To evaluate the approach which is detailed in this Thesis Dissertation, in this chapter a wide empirical study over a real scenario is performed. This way, this chapter includes the description of the real scenario where the evaluation took place (cf. Section 5.2) as well as the case study which was followed to carry out such evaluation (cf. Section 5.3).

5.2 A Real Example: A Beauty Salon of Seville

This section introduces a real example related to a beauty salon that is used to validate the proposed approach in the considered case study. Specifically, in Section 5.2.1 the selection of the considered scenario for carrying out the empirical evaluation is motivated, Section 5.2.2 explains how the proposed approach is intended to be used for improving the current situation, Section 5.2.3 details the considered scenario, Section 5.2.4 includes the SDeclare specification of the considered scenario, and, finally, Section 5.2.5 explains how the proposed approach can be used in the considered scenario.

5.2.1 Motivation

The considered business has grown considerably in the last years. It has expanded from a small salon with three employees to more than six and included additional facilities to be able to offer additional services. In addition, the uncertainty regarding different aspects of the business has become an important problem, e.g., the arrival time of the clients or the availability of some resources during the day (e.g., due to a resource who feels sick at the beginning of the day but not enough

to leave the salon). These changes, including the quick growth together with the complex constraints which need to be obeyed, resulted in problems related to the management of the salon. In particular, long waiting time for clients and missing schedules for employees are causing problems, affecting customer satisfaction and profit of the business.

Furthermore, such scenario has been selected since:

1. It faces a complex problem which goes far beyond a toy problem since it presents several challenges that can also be found in other domains (i.e., it highly depends on the managers skill to take decisions, uncertainty is inherit to the business and the declarative language ameliorate it specification).

2. Unlike more common scenarios, this kind of business has not been widely supported by previous research and thus, it is considered an innovative application.

3. Access to the data was possible in that scenario and therefore, the analyst was able to collect data for a long period of time for the evaluation.

5.2.2 Goal of the Business

The goal of the business is to improve the current situation through the optimization of some business objective functions. Since our approach generates an optimized configurable BP model, a set of optimized schedules for employees can be suggested each one facing a different possible uncertain scenario, and therefore, the aforementioned problems can be overcome. Moreover, since multi-objective optimization is considered, several important objectives (i.e., minimizing waiting times for clients and maximizing profit) can be optimized. Furthermore, due to the high expressiveness of SDeclare, all the constraints which are given in the scenario together with its uncertainty can be specified.

5.2.3 Scenario details

The beauty salon offers various services [1] like dye, clean&cut, manicure and facial services. It requires its clients to make appointment calls to know how many clients are coming as well as the booked services. There are several full-time employees, e.g., Amparo (A), Rosa (R), Lisset (L) and Marta (M). Each employee has different skills, and hence some activities can be performed by certain employees only. For all activities which are performed in the salon, the manager

[1] For the sake of clarity, the depicted scenario is a subset of the actual beauty salon, i.e., the salon offers more services and has more employees.

knows the average estimated duration, the profit which is obtained after their execution, and the employees which can execute that activity. The manager of the salon wants to plan and schedule a working day with several clients taking the following considerations into account:

1. The profit (P) of the resulting working plan has to be maximized (objective function 1).
2. The waiting time (WT) of the clients has to be minimized and distributed uniformly among all the clients (objective function 2):
   $WT = \frac{\sqrt{\sum_{c \in C}((s.endT(c) - c.appT) - (\sum_{b \in c.served} b.estimate))^2}}{C.size}$, where C is the set of clients, s is the considered solution, $s.endT(c)$ is the time when the client c has finished, $c.appT$ is the appointment time of c, $c.served$ is the set of services which are applied to c (i.e., included in the plan), and $b.estimate$ is the estimated duration for service b.
3. The employees can offer some additional services to the client directly in the salon, and the client can accept or refuse. However, these additional services should only be proposed if this leads to optimized plans.

5.2.4 SDeclare Specification

Typically, as illustrated in Figure 5.1, a client visit starts with the reception in the beauty salon. After that, the staff applies some services to the client and, finally, the client is charged. Complex activity *Services* is composed of other activities[2] (e.g., dye, clean&cut, facial and manicure, cf. Figure 5.2), while *Reception* and *Charge* are S-Activities (cf. Definition 24). For each S-Activity two attributes are considered: (1) estimated activity duration, and (2) profit which is obtained after executing the activity.[3] Moreover, the set of alternative resources which can perform the S-Activity is also included (cf. Example 28).

Example 28. *In Figure 5.1, activity Reception has an estimated duration of 1 minute and a profit of 0, and can be performed by A, R, M or L.*

Notice that each instance created from the model of Figure 5.1 represents one client visiting the beauty salon. The current problem deals with N clients (represented by the Existence constraint of Figure 5.1, stated by the label N) which come to the salon at different times and with different bookings during a working day which are specified as data information.

[2] In a similar way to PSL (PSL, 1977), SDeclare allows hierarchical modelling (i.e., complex activities aggregate activities).

[3] As can be seen in Figs. 5.1 and 5.2, the profit of the services is associated to one of the activities of the related services.

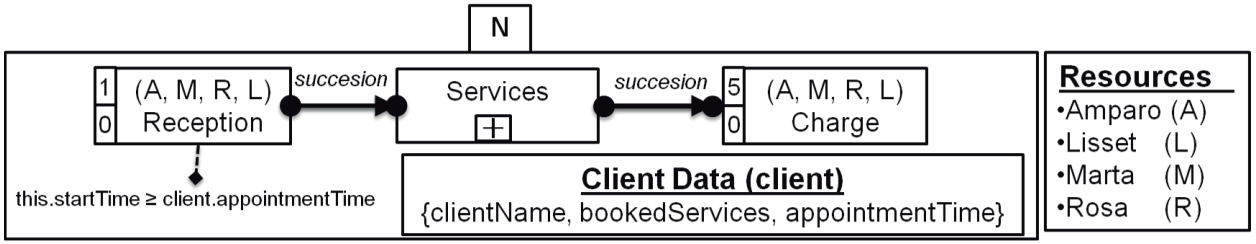

Figure 5.1: SDeclare Model for the Beauty Salon Problem (Top level process)

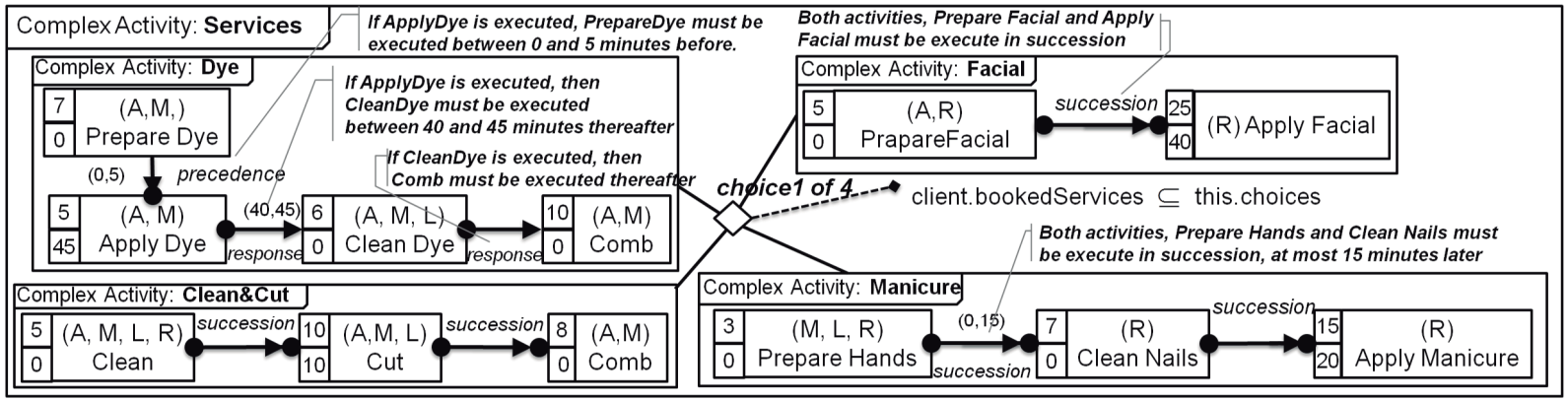

Figure 5.2: SDeclare Model for some of the Services which are offered

The data perspective also appears in Figure 5.1. The *Client-Data* includes all the information which is related to the client bookings, and consists of: (1) *clientName*, (2) *bookedServices*, which represents the mandatory services that the salon staff has to cover, and (3) *appointmentTime*, which is the time when the client is supposed to arrive at the salon. Through the data perspective, it is possible to model that activity *Reception* cannot start before the client appointment time (cf. Figure 5.1). Moreover, a data constraint is used (in conjunction with the choice constraint) to ensure that all the services the client has booked are selected, i.e., the generated plans will always include the booked services (cf. Figure 5.2).

5.2.5 Applying the Proposed Approach

Given a SDeclare model $SDM = (Acts, Data, C_{BP}, AvRes, OFs)$ for the beauty salon problem, where *Acts*, *Data*, C_{BP} and *AvRes* are shown in Figs. 5.1 and 5.2, and *OFs* are described in scenario details (i.e., maximization of the profit and minimization of the waiting time), the proposed tool generates multi-objective optimized enactment plans. These plans are, in turn, represented as a configurable BP model by following one of the available policies. Such model will support the manager of the beauty salon in managing the working day in an optimized way. In addition, our approach will guide the individualization of the configurable BP Model through questionnaires.

In this way, the proposed approach provides support to the manager of the beauty salon by suggesting (cf. Example 29): (1) a resource for executing each activity, (2) the start and end time of the activities, and (3) the services which will be offered to each client (i.e., services which were not booked by the client).

Example 29. *As examples, two Pareto optimal plans for serving 10 clients in the beauty salon are depicted in Figure 5.3 and Figure 5.4.*

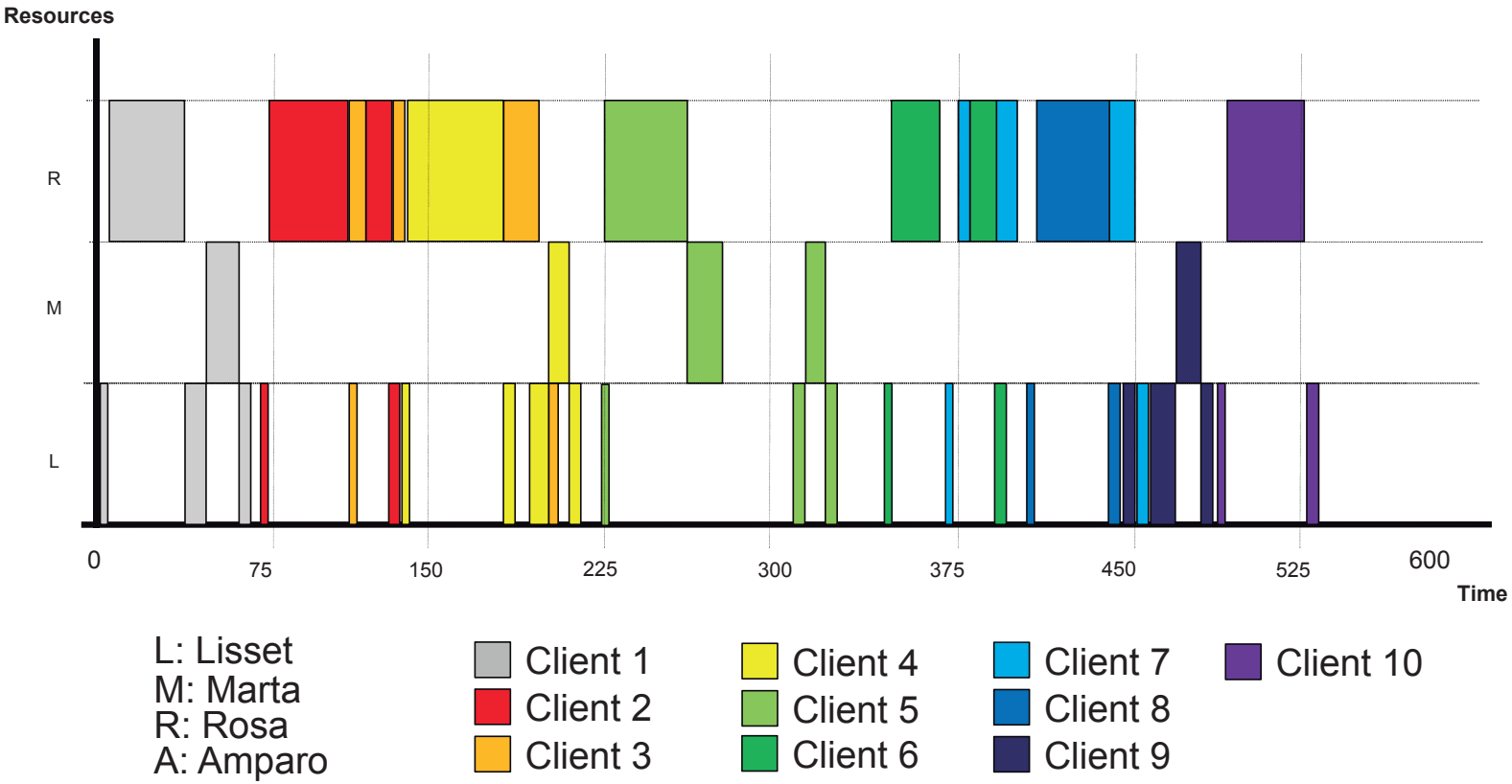

Figure 5.3: Example of an enactment plans for the beauty salon problem when using 3 resources.

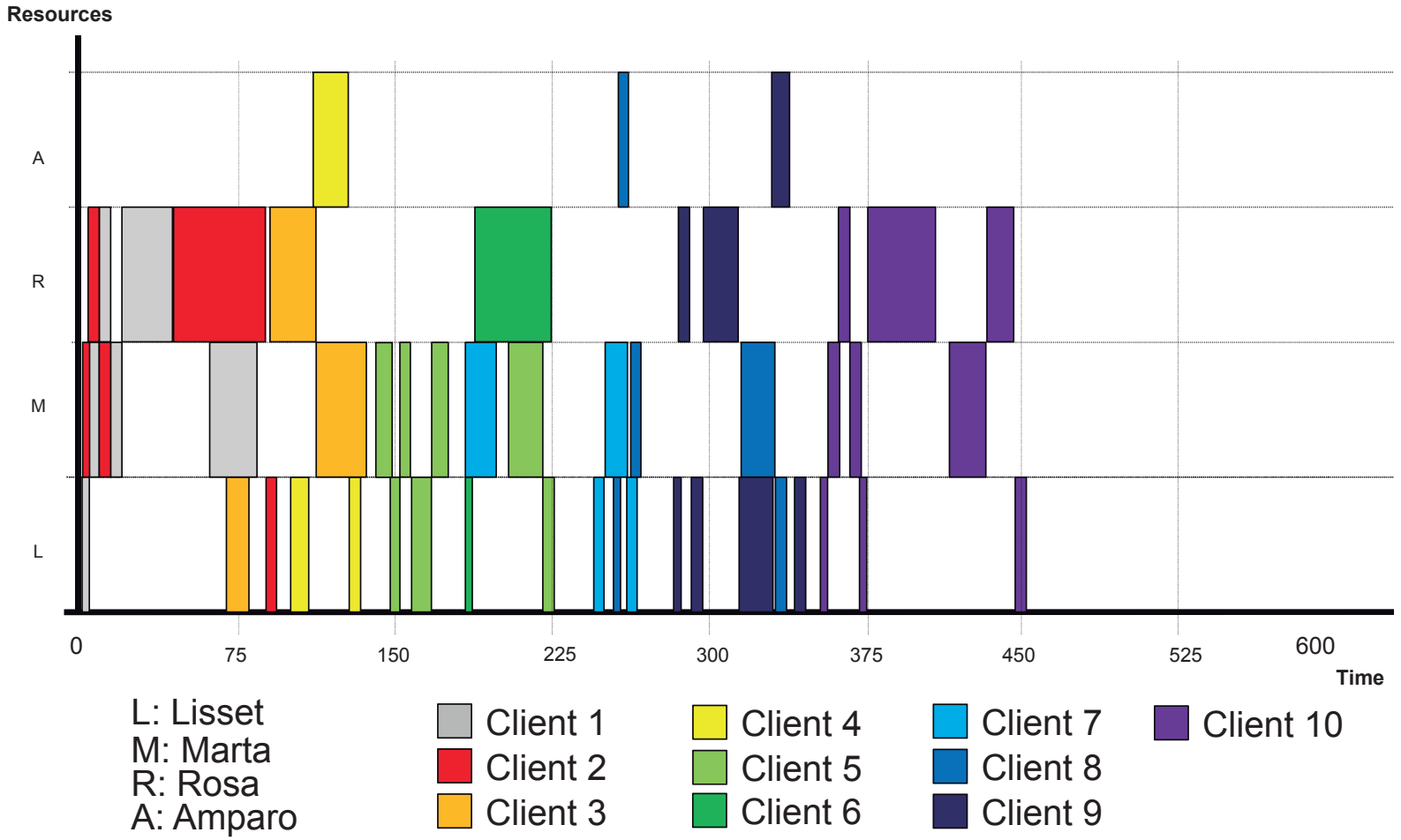

Figure 5.4: Example of an enactment plans for the beauty salon problem when using 4 resources.

5.3 Case Study

In this section, the case study protocol for the software engineering field proposed by (Brereton et al., 2008) is followed to improve the rigour and validity of the study. Such protocol suggests the following sections: background, design, case selection, case study procedure, data collection, analysis and interpretation, and validity evaluation.

5.3.1 Background

In this section, previous research related to the topic of this Thesis is identified. Different proposals related to (1) the generation of optimized BP enactment plans and the related optimized configurable BP models from declarative specifications, (2) flexibility and robustness concerns, and (3) the generation of questionnaires for individualizing configurable BP models are discussed in the previous chapters (cf.). In particular, our proposal uses the SDeclare language for the declarative specification of a BP and generates a set of optimized enactment plans out of it. After that, a configurable BP model is created using those plans which more contribute to the flexibility and robustness. Thereafter, a questionnaire is automatically created for individualizing the generated configurable BP model in run-time.

In such context, the *purpose of this study* is the evaluation of the proposed approach for guiding the execution of declarative models through automatically-generated questionnaires. Taking the purpose of the study into account, three main research question (MQs) are defined (cf. Table 5.1) as follows:

1. *MQ*1 checks the suitability of Alg. 1, i.e., evaluates if the obtained optimized BP enactment plans are uniformly distributed over the solution space[4] and if the algorithm performs well when dealing with complex problems. For this, *MQ*1 is divided in two additional research questions (AQs):

   (a) *AQ*1 checks whether Alg. 1 finds solutions within the different regions in which the solution space is divided.

   (b) *AQ*2 evaluates if Alg. 1 behaves successfully (i.e., finds a uniformly distributed set of solutions) when solving problems of different complexity.

2. *MQ*2 assesses if the current approach can be useful to deal with real problems involving uncertainty. For this, *MQ*2 is divided into two additional questions:

[4]Note that one of the goals of Alg. 1 consists of obtaining an uniformly distributed set of solutions.

   (a) *AQ*3 checks whether the solutions which are obtained by our approach improves the actual solutions which are manually obtained by the domain expert for the real scenario in terms of their objective functions.

   (b) *AQ*4 evaluates the generated configurable BP models in terms of its robustness and flexibility against the input uncertainty, which is important in order to avoid replanning.

3. *MQ*3 evaluates the proposed method for automatically creating questionnaires for configurable BP models as well as the method for incrementally configuring them. For this, *MQ*3 is divided in four additional questions:

   (a) *AQ*5 checks if the approach behaves properly against configurable nodes of different sizes.

   (b) *AQ*6 evaluates the suitability of the questions which are generated.

   (c) *AQ*7 is concerned about the performance which is achieved using the current approach.

   (d) *AQ*8 checks if replanning can be avoided.

5.3.2 Design

The *object of study* is the approach which is proposed for guiding the executions of declarative models. For this, three different designs are carried out in this case study.

1. A first *embedded* design (*ED*1 in the following) concerning Alg. 1. Particularly, this first design considers one analysis unit: the generation of optimized BP enactment plans through solving MO-COPs. In this design, for addressing *MQ*1 (i.e., *AQ*1 and *AQ*2), a set of different non-stochastic SDeclare models are randomly generated leading to different complexities of the MO-COPs which have to be solved. Note that stochastic variables are not considered in this design.

2. A second *embedded* design (*ED*2 in the following) which considers the generation of configurable BP models from declarative specifications for addressing *AQ*3 and *AQ*4. Specifically, the method is applied over different SDeclare models in which stochastic variables are considered for the appointment time of the clients and for the availability of resources. For dealing with such variables, the sampling step is configured to generate 30 different samples (cf. Section 4.2.1).

Table 5.1: Case study research questions

Id	Research Question
MQ1	Is Alg. 1 appropriate for finding a uniformly distributed set of Pareto optimized solutions for SDeclare models of different complexity?
AQ1	Can Alg. 1 find solutions within the different regions in which the solution space is divided?
AQ2	Does Alg. 1 behave successfully independently of the complexity of the problems?
MQ2	Is the proposed approach useful for a business expert?
AQ3	Can the proposed approach improve the results which are manually obtained by an expert?
AQ4	Can the proposed approach generate an unified artifact which behaves properly against the input uncertainty?
MQ3	Is our method appropriate for individualizing configurable BP models during run-time?
AQ5	Can the proposed method be used to generate questions for configurable nodes of different sizes (i.e., nodes with different number of branches)?
AQ6	Are the generated questionnaires appropriate to be answered in a real environment (i.e., adequate number of questions)?
AQ7	Is the business performance improved by using the proposed method?
AQ8	Is the proposed method preventing replanning (i.e., changing the variant which is being executed)?

3. A third *embedded* design ($ED3$ in the following) considering the creation of questions for addressing $AQ5$ and $AQ6$. In this design, the method for generating questionnaires from a configurable node (cf. Section 4.3) is considered as the analysis unit. For this, such method is applied over a set of configurable nodes of different sizes.

4. A forth *embedded* design ($ED4$ in the following) which considers the configuration of configurable BP models using questionnaires for answering $AQ7$ and $AQ8$. Specifically, the proposed approach is applied over different configurable BP models each one presenting a different complexity (i.e., different number of activities).

For addressing the design $ED1$ and $ED2$, Alg. 1 considers dividing the solution space in 4 regions rx ($rx \in \{r1, r2, r3, r4\}$) and the constraint-based search algorithm is run until a 5-minutes CPU $TIME\_LIMIT$ is reached (cf. Algs. 2 and 3), which is considered a reasonable amount of time for this business.

Table 5.2: Quantified variables for $ED1$

Variable	Description
$minWT$, $minP$, $MaxWT$, $MaxP$	Initial ranges for each objective function, which are calculated in the initial searches of Alg. 1.
$\%SF_{rx}$	The percentage of SDeclare models in which, at least one solution is found within region region rx ($rx \in \{r1, r2, r3, r4\}$) regarding the total number of SDeclare models which are considered.
$NActs_{rx}$	Size of the enactment plans which are generated within region rx, which is measured as the number of activities.
WT_{rx}, P_{rx}	Value for each objective function which is obtained within region rx.
$SAdd_{rx}$	Number of additional services per client which are included in the solutions which are found within region rx (i.e., those services which where not initially booked but are included in the plan)
$\%PS_{rx}$	Percentage of Pareto optimized solutions obtained within region rx regarding the total number of Pareto optimized solutions which are obtained for the considered problem

All the aforementioned designs are run on a Intel(R) Xeon(R) CPU E5530, 2.40GHz, 8GB memory, running Debian 6.0.3. After carrying out the four designs, the generated information (i.e., optimized BP enactment plans and questionnaires) is analyzed to answer the research questions (cf. Table 5.1).

As follows, the data which is quantified for each design is detailed. First, for $ED1$, the data described in Table 5.2 is quantified (cf. Example 30) for each SDeclare model which is considered.

Example 30. *Figure 5.5 depicts the set of solutions which are found during the search process for a specific problem. As can be seen, Alg. 1 divides the solution space in 4 regions. In order to state the limits of each region (i.e., minWT, MaxWT, minP and MaxP), the solutions which are found in the first step of Alg. 1 (cf. Alg. 2) are used (depicted by squares in Figure 5.5).*

In this example, as only one problem is considered, $\%SF_{r1} = \%SF_{r2} = \%SF_{r3} =$ 100% *since at least one solution is found within* $r1$, $r2$ *and* $r3$, *and* $\%SF_{r4} = 0\%$. *As stated in Alg. 3, since a solution is found in* $r1$ *and it dominates* $r4$, $r4$ *is not explored. Note that the solutions which are depicted within* $r4$ *were obtained in the first step.*

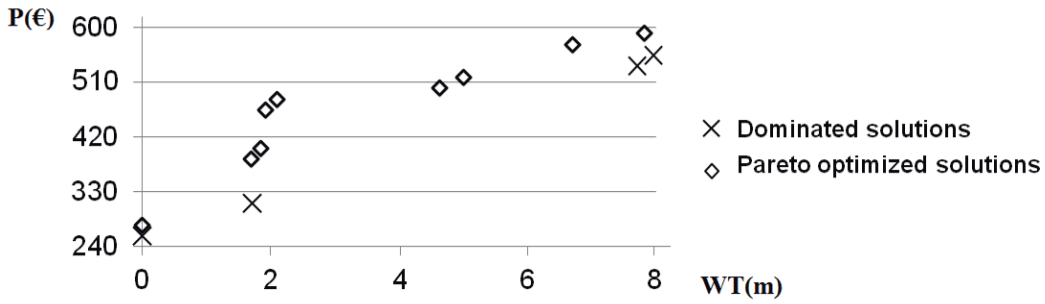

Figure 5.5: Solutions which are found for a specific setting of the beauty salon problem

Within the overall solution space, 22 different solutions are found. Of these 22 solutions, 9 are Pareto optimized while 13 are dominated (cf. Definition 7). From the 9 Pareto optimized solutions, 2 solutions belong to $r1$*, 4 to* $r2$ *and 3 to* $r3$*, which means that* $\%PS_{r1} = 22.2\%$*,* $\%PS_{r2} = 44.4\%$ *and* $\%PS_{r3} = 33.3\%$*. As* $r4$ *is not solved, i.e.,*$\%PS_{r4} = 0\%$*.*

For the $ED2$ design, the data described in Table 5.3 is quantified (cf. Example 31) for the different non-stochastic SDeclare models which are generated for each considered SDeclare model. In order to measure $\overline{Flex}$, *MaxRob* and *minRob*, all plans which are generated for each non-stochastic SDeclare model are kept, i.e., the "all plans" policy (cf. Section 4.2.4) is applied.

Example 31. *Figure 5.6 shows the set of solutions which are obtained when solving a specific problem related to the beauty salon scenario. As can be seen, many Pareto optimized solutions (i.e., squares in Figure 5.6) are obtained since the problem is solved several times because of the uncertainty variables (i.e., different samples are generated and solved, cf. Section 4.2.1). The maximum and minimum values (i.e.,* $minWT=3$*,* $MaxWT=18$*,* $minP=350$ *and* $MaxP=730$*) as well as their average values (i.e.,* $\overline{WT}=10.3$ *and* $\overline{P}=601$*) are depicted in each axis.*

In addition, the real execution plan is depicted by a circle in Figure 5.6. The difference between $\overline{WT}$ *(*$\overline{P}$*) and* $WT=14.8$ *(*$P=515$*) in the real execution plan is equal to* $\Delta WT=4.5$ *(*$\Delta P=86$*). Therefore, applying the proposed approach, the waiting time has been reduced (i.e.,* $\%\Delta WT=30.4\%$*) in average and the profit has been incremented (i.e.,* $\%\Delta P=16.7\%$*) in average.*

Finally, as depicted, the real enactment plan is dominated by some Pareto optimized solutions (cf., arrows in Figure 5.6) which means that solutions which improve both objective functions are found. Therefore, as only one problem is considered in this example, $\%Dominated = 100\%$*.*

For the design $ED3$, the data described in Table 5.4 is quantified for each configurable node. In addition, the business manager specified that answering

Table 5.3: Quantified variables for $ED2$

Variable	Description
$minWT$, $minP$, $MaxWT$, $MaxP$	Average of the minimum and maximum values of the objective functions (i.e., waiting time and profit) which are obtained by applying the proposed approach.
$\overline{WT}$, $\overline{P}$	Average values of the objective functions which is obtained applying the proposed approach.
$\%\Delta WT$, $\%\Delta P$	Average values of the percentage of increment of the objective functions of the plans which are obtained through the proposed approach versus the real execution plan.
$\%Dominated$	Percentage of problems whose real execution plan is dominated (cf. Definition 6) by the Pareto front generated by our approach.
$\overline{Flex}$	Average flexibility (cf. Definition 38) of the generated configurable BP models against the uncertainty provided by the input SDeclare model.
$minRob$, $MaxRob$	Average values of the minimum and maximum robustness (cf. Definition 37) of the plans which are included in the generated configurable BP models.

more than 10 questions would be inefficient and thus, $AQ2$ can be answered as true if $\#MQ$ stays under 10 independently of the size of the configurable node.

Lastly, for the design $ED4$, the data described in Table 5.5 is quantified for each configurable BP model.

5.3.3 Case Selection

For this case study, the beauty salon problem is studied. This is considered a good and suitable case since it fulfills the following selection criteria: (1) it has been created for **an actual business**, (2) the business has grown up and now it has **scheduling problems** (i.e., involves resource allocation, complex constraints and multi-objective optimization), (3) the business performance **highly relies on runtime decisions** (i.e., the knowledge of the domain expert has a great influence on the performance), and (4) the problem is **subject to uncertainty**, and such uncertainty can be measured, i.e., the manager can detect the uncertainty of the scenario which can be manually specified in the SDeclare models.

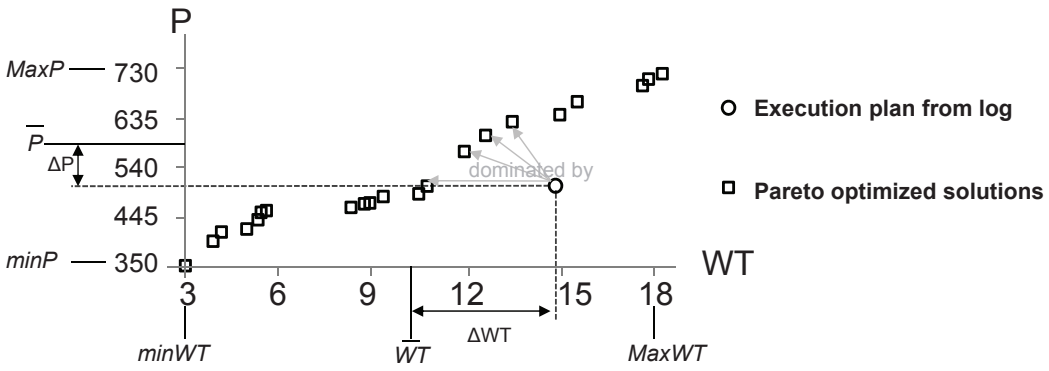

Figure 5.6: Solutions which are found for a specific problem related to the beauty salon problem with some stochastic variables

Table 5.4: Quantified variables for $ED3$

Variable	Description
OB	The number of outgoing branches .
#MQ, *#mQ*	The minimum and maximum number of questions which need to be answered for resolving the questionnaire associated to such node.

5.3.4 Case Study Procedure

The execution of the study is planned as follows.

1. The business is selected according to the selection criteria.

2. The selected business is modeled as a SDeclare model by the business analyst. Initially, only the activities and the constraints which relate them are included.

3. A different procedure is performed depending on the case design which is being carrying out:

   - In the case of the $ED1$ design, different data for the SDeclare model of the Beauty Salon problem (cf. Section 5.2) are randomly generated. Therefore, each generated model includes the same activities, relations and resources, but differs in the number of clients (N), their booked services (S), and their appointment times (T). Considering the information which is provided by the manager of the salon (i.e., there are normally between 10 and 20 clients per day and a client typically books one or two services) values $\{1, 1.5, 2\}$ are considered for the

Table 5.5: Quantified variables for $ED4$

Variable	Description
$\#Acts$	The number of activities of the configurable BP model.
$\#Q$	The number of questions which the user actually answers for individualizing the configurable BP model.
$\Delta\$$	The increment of profit which is obtained by using the current approach versus not using it.
$\%R$	The percentage of cases in which replanning is avoided by using the current approach.

average number of booked services of the clients (i.e., NS) and the values $\{10, 15, 20\}$ for the number of clients (i.e., N). Based on this information, to average the results over a collection of randomly generated SDeclare models, 30 data instances are randomly generated for each pair $<N, NS>$ by varying S and T.[5]

After that, the optimized BP enactment plans are obtained by applying Alg. 1.

- In the other cases (i.e., $ED2$, $ED3$, and $ED4$ designs), the model of Figure 5.1 has been extended to reflect the reality better, i.e., the real model includes 21 different services and 7 resources. In addition, some services are related to each other to prevent non-sense executions, e.g., to avoid performing the cleaning nails service after the painting nails service. Furthermore, the number of services per client is limited to 4.

  In addition, for the $ED3$ and $ED4$ designs, the salon manager provided a set of properties in form of functions (i.e., the well-defined properties written in the business language) which can be calculated from each enactment plan.

  As mentioned, this case design considers real data which is obtained from the log of the beauty salon. For this, the staff of the beauty salon manually logged data for a period of 90 days. In particular, for each day they logged: (1) the number of clients (i.e., N), (2) their booked services (i.e., S), (3) their appointment times (i.e., T), and (4) the resource availability of each day ($AvRes$). In addition, for each event that occurs during the day (e.g., when a client arrives, an activity starts

[5]The set of problems which are used for the empirical evaluation is available at http://regula.lsi.us.es/MOPlanner/ObjectsBeautySalon.zip.

or finishes), its time-stamp is recorded by the receptionist. In turn, the manager of the beauty salon provided some stochastic variables related to T (i.e., some unpunctual clients) and to *AvRes* (i.e., staff who probably become unavailable during the day, cf. Example 32). These stochastic variables were defined with flat PMFs (cf. Section 4.1).

Example 32. *Let 14:30 be the appointment time of certain client. The manager knows that such client used to be unpunctual; therefore, she specifies in the SDeclare model her appointment time through the flat PMF appT= {14:25, 14:30, 14:35, 14:40}.*[6] *In addition, before starting the day (and before generating the configurable BP model), one of the resources has informed the manager that she does not feel well; therefore, the manager specifies her availability through the flat PMF avR={0 ,1} in the SDeclare model.*

Therefore, for each day, the same SDeclare model is considered (i.e., they have the same activities, relations and estimates) but each day differs in N, S, T, *AvRes* and the associated stochastic variables (i.e., problem data). [7]

These problems are grouped considering N and the average number of booked services (i.e., $NS = |S|/N$) in order to enable the comparison with the $ED1$ design. For this, N is divided into three ranges $[8,12]$, $[13,17]$ and $[18,22]$, and NS is also divided into three ranges, i.e., $[1,1.4]$, $(1.4,1.8]$, $(1.8,2.2]$.

Then, the configurable BP models are obtained by applying the proposed approach. As mentioned, only the best variants are kept when generating the configurable BP model.

In addition, for 30 of those 90 days, the salon manager was supported by our tool to manage configurable BP models for the $ED3$ and $ED4$ designs. Using such a tool, after the configurable BP model is generated for each day, the variant which was selected by the salon manager before starting the execution (i.e., before the first client arrived) is logged. In addition, each time a configurable node appears (i.e., a decision needs to be taken), a questionnaire was prompted and she answered it. At the end of each day, the #Q and the selected variant (i.e., the result of the individualization) were stored. In addition, such variant was compared with the variant selected before starting the execution and Δ\$ was calculated and stored for each day. Furthermore, the

[6] In this example, 5 minutes is considered as the minimum amount of time which can be measured.

[7] The set of data logged which is used for this experiment is available at http://regula.lsi.us.es/MOPlanner/ObjectsBeautySalonReal.zip.

variant which was selected before starting the execution was checked if could withstand the events logged for that day. In case it would not, it was stored if replanning was avoided by using our approach, i.e., $\%R$ is stored.[8] The value for $\%R$ is calculated as the percentage of times that our approach avoided replanning against the total number of times that replanning was required. To analyze the behavior of the method against different complexities, the 30 configurable BP models (each one corresponding to a day of work) are grouped considering $\#Q$. In particular 4 groups are considered: $\#Q \in [40,60)$, $\#Q \in [60,80)$, $\#Q \in [80,100)$, and $\#Q \in [100,120)$.

In turn, after the period of 30 days passed, for the $ED3$ design, all the configurable nodes which appeared in the configurable BP models which were stored were gathered. Specifically, 259 configurable nodes were obtained and the current approach was applied to generate the questionnaire associated to each node. For each node, OB, $\#mQ$ and $\#MQ$ were stored. To analyze the behavior of the method against different complexities, the 259 configurable nodes were grouped considering OB. In particular 4 groups were considered: $OB \in [2,5)$, $OB \in [5,8)$, $OB \in [8,11)$ and $OB \in [11,14)$.

4. All the relevant information is collected following the collection plan.

5. Finally, the analysis and the interpretation of the collected data is conducted and the validity of the case study procedure is studied.

5.3.5 Data Collection

Different data collection plans are conducted depending on the case study design.

1. In the $ED1$ design, for each pair $<N,NS>$, the data related to the quantified variables (cf. Table 5.2) is collected in three phases while generating the optimized BP enactment plans from the SDeclare models. Such phases are detailed as follows. (1) After the initial searches are performed (i.e., Alg. 2 is executed, the solution space is divided into four regions, and the values of the variables $minWT$, $MaxWT$, $minP$ and $MaxP$ are recorded). (2) After such division, $r1$ is the first region to be solved (note that, as mentioned, $r4$ is dominated by $r1$, cf. Section 3.2.2 and Example 30). Then, the data related to $\%SF_{r1}$, WT_{r1}, P_{r1}, $NActs_{r1}$, $\%PS_{r1}$, and $SAdd_{r1}$ is stored. (3)

[8] Note that cases in which replanning becomes necessary may exist although run-time configuration is applied. In such situations, a new configurable BP model is created ensuring that all the included variants cover the given situation as discussed in (Barba et al., 2013a,b).

Table 5.6: Quantified variables for the $ED1$ design (1)

Problem		Unconstrained			
N	NS	minWT	MaxWT	minP	MaxP
10	1	0.1	29.0	273.3	660.0
10	1.5	1.0	31.5	350.5	793.4
10	2	2.1	36.3	492.7	956.3
15	1	3.0	32.4	360.3	782.0
15	1.5	9.7	43.4	538.4	891.5
15	2	8.0	49.6	741.8	1092.1
20	1	15.5	56.3	405.6	859.1
20	1.5	17.8	59.5	715.6	1108.0
20	2	16.5	59.4	950.6	1252.8

Finally, $r2$ and $r3$ are solved. In the case that no solution is found in $r1$, $r4$ is also solved. Similarly to the previous phase $\%SF_{rx}$, WT_{rx}, P_{rx}, $NActs_{rx}$, $\%PS_{rx}$, and $SAdd_{rx}$ are recorded for such regions. Tables 5.6, 5.8, 5.7, 5.9, and 5.10 show the values related to all the quantified variables which are involved in the complete data collection plan.

2. In the $ED2$ design, for each pair $<N, NS>$, once the configurable BP model is generated, the data related to the quantified variables (cf. Table 5.3) is recorded, i.e., $minWT$, $MaxWT$, $minP$, $MaxP$, $\overline{WT}$, $\overline{P}$, $\%\Delta WT$, $\%\Delta P$, $\%Dominated$, $\overline{Flex}$, $minRob$, and $MaxRob$. The aforementioned values are shown in Tables 5.11 and 5.12.

3. In $ED3$ design, after the period of 30 days has passed, the information related to each OB is quantified, i.e., $\#mQ$ and $\#MQ$ (cf. Table 5.4). Such values are shown in Table 5.13.

4. In $ED4$ design, at the end of each day, the quantified variables of this design (i.e., $\#Q$, $\Delta\$$, and $\%R$, cf. Table 5.5) are calculated and associated to the $\#Acts$ of this day. Table 5.14 shows the values which are obtained.

5.3.6 Analysis and Interpretation

The data which is collected is analyzed to answer each research questions and to draw conclusions, as detailed as follows:

1. In order to address question $MQ1$, sub-questions $AQ1$ and $AQ2$ need to be answered (cf. Tables 5.6, 5.8, 5.7, 5.9, and 5.10). Regarding the problems

Table 5.7: Quantified variables for the $ED1$ design (2)

Problem		Region 1					
N	NS	$\%SF_{r1}$	WT_{r1}(m)	P_{r1}(€)	$NActs_{r1}$	$\%PS_{r1}$	$SAdd_{r1}$
10	1	74.2	14.3	479.0	90.1	10.0	2.5
10	1.5	63.8	15.2	491.0	98.3	9.2	1.5
10	2	67.0	17.6	615.2	145.6	11.5	1.8
15	1	70.5	15.8	515.0	116.9	11.1	2.1
15	1.5	61.4	22.3	698.4	154.8	9.6	1.4
15	2	64.9	24.2	918.4	179.3	9.5	0.9
20	1	72.6	29.6	625.0	150.5	10.4	1.2
20	1.5	59.9	30.9	859.7	172.1	10.5	0.6
20	2	45.2	30.1	1105.2	196.0	8.9	1.0

which are solved within each region (i.e., columns $\%SF_{rx}$), $\%SF_{r1}$ is lower than both $\%SF_{r2}$ and $\%SF_{r3}$ since $r1$ is the most constrained region. In turn, the low value for $\%SF_{r4}$ can be explained by the fact that solutions are not searched in $r4$ in the case that at least one solution is found in $r1$. In addition, as the complexity of the problem increases (i.e., N and NS increase), $\%SF_{r1}$ decreases and therefore, $\%SF_{r4}$ increases. However, $\%SF_{r2}$ and $\%SF_{r3}$ keep similar values and close to 100%, which means that at least one solution can be found in $r2$ and $r3$ regardless of the complexity of the problems. Therefore, $AQ1$ can be answered as true as solutions can be found in all regions.

Furthermore, some differences can be observed when analyzing the objective function values (i.e., columns WT and P). As expected, as the complexity of the problems increases, the value of P increases since more services are included in the generated enactment plans. Moreover, with increasing complexity of the problem the value of WT increases as well since the clients are subject to more delays. Note that these columns are also directly dependent on the number of activities of the plan (cf. columns NActs), i.e., when NActs increases, P and WT increase too. In general, $r1$ and $r4$ include the most balanced solutions according to the values of both objective functions, while the solutions with the best values for P and WT belong to $r2$ and $r3$ respectively. This distribution is kept independently of the complexity of the problem. Moreover, for all problems Pareto optimized solutions were obtained (cf. columns $\%PS_{rx}$), which means that a representative Pareto front can be depicted. However, $r4$ contributes less to the Pareto front since most of the solutions which were found within $r4$ are

Table 5.8: Quantified variables for the $ED1$ design (3)

Problem		Region 2					
N	NS	$\%SF_{r2}$	WT_{r2}(m)	P_{r2}(€)	$NActs_{r2}$	$\%PS_{r2}$	$SAdd_{r2}$
10	1	100.0	19.6	502.4	118.6	38.2	2.7
10	1.5	100.0	26.2	517.4	128.4	43.5	1.9
10	2	100.0	26.9	694.8	157.5	44.0	2.2
15	1	100.0	29.6	521.0	130.0	42.1	2.6
15	1.5	100.0	34.5	712.4	166.0	39.6	1.8
15	2	96.7	39.7	982.4	181.7	42.3	1.4
20	1	100.0	39.0	621.5	139.5	40.8	1.2
20	1.5	96.2	42.9	902.8	180.6	39.1	0.9
20	2	96.2	49.0	1193.6	201.1	47.9	1.1

dominated by solutions found within $r2$ or $r3$. In turn, $\%PS_{r1}$ also presents low values since $r1$ is the hardest region to be solved (i.e., the most constrained). Furthermore, in all regions $\%PS_{rx}$ seems to be independent of the complexity of the problem. Therefore, although the time spent by Alg. 1 is only based on the $TIME\_LIMIT$ constant, it behaves as intended against all the different complexities (i.e., $\%SF_{r2}$, $\%SF_{r3}$ and columns $\%PS_{rx}$ are independent on the complexity of the problems, while WT_{rx} and P_{rx} values are directly dependent on the complexity of the problems), and hence, $AQ2$ can be answered as true.

Finally, for each problem, a relation among the number of additional services per client (cf. columns SAdd), P and WT exists. Specifically, as the number of services which are included increases, the profit also increases to the detriment of the waiting time. However, as the problems become more complex, SAdd decreases in all the regions since it is more complicated to include more services since more clients and services have to be considered. Considering these values and that $AQ1$ and $AQ2$ are answered as true, $MQ1$ is concluded as true, i.e., the proposed algorithm is suitable for generating a distributed set of Pareto optimized solutions starting from a SDeclare model.

2. In order to address question $MQ2$, sub-questions $AQ3$ and $AQ4$ need to be answered (cf. Tables 5.11 and 5.12). As can be seen, the column $\overline{WT}$ shows that the solutions provided by our approach are shifted to the lowest part of the range [minWT, MaxWT]. This means that more solutions were found in the region related to that part than in the other regions. In turn, the values of

Table 5.9: Quantified variables for the $ED1$ design (3)

Problem		Region 3					
N	NS	$\%SF_{r3}$	WT_{r3}(m)	P_{r3}(€)	$NActs_{r3}$	$\%PS_{r3}$	$SAdd_{r3}$
10	1	100.0	17.0	340.6	79.5	49.3	1.1
10	1.5	100.0	14.5	365.5	80.1	45.4	0.9
10	2	97.0	15.8	511.3	109.7	42.4	1.4
15	1	100.0	15.1	394.2	83.4	45.3	1.1
15	1.5	96.7	22.1	566.3	129.1	50.1	0.7
15	2	100.0	21.5	735.0	162.0	44.9	0.9
20	1	95.0	26.3	498.7	99.9	46.6	0.8
20	1.5	96.2	27.6	725.9	160.4	46.4	0.7
20	2	100.0	28.4	997.2	191.8	39.1	0.6

$\overline{P}$ are more balanced though still being within the lowest part of the range [minP, MaxP]. To overcome this issue, the solution space can be divided in more regions in order to get more balanced solutions. Moreover, $\%\Delta WT$ increases as the complexity of the problems increases, which highlights the benefits of using the proposed approach in real cases. The opposite happens with $\%\Delta P$ since the more complex the problem is, the fewer free time slots are available to offer more services. Nonetheless, in all the cases the values $\%\Delta WT$ and $\%\Delta P$ show that our approach improves the mean of both objective functions compared to plans which were manually created (i.e., WT decreases and P increases). Moreover, the solutions which are provided by our approach dominate the associated real plan in all the cases (cf. column $\%Dominated$). That means that, regarding these objective functions, Alg. 1 provides at least one solution which improves both profit and waiting time when compared with the real solutions. Therefore, $AQ3$ is answered as true.

Regarding the flexibility of the generated configurable BP models (cf. column $\overline{Flex}$), in most cases it achieves 100%, which means that the uncertainty which was specified by the manager is totally covered by the generated models. In fact, the value of $\overline{Flex}$ is over 83.3% even in the most complex problem, which represents a very high degree of flexibility. Regarding the robustness, as the complexity of the problems increases, both upper and lower limits of the robustness decrease (cf. columns minRob and MaxRob). This is due to the fact that the more activities exist in the plan, the less slack appears. However, these columns present rather good values since a value of 13.3% of robustness means that the related plan will avoid

Table 5.10: Quantified variables for the $ED1$ design (5)

Problem		Region 4					
N	NS	$\%SF_{r4}$	WT_{r4}(m)	P_{r4}(€)	$NActs_{r4}$	$\%PS_{r4}$	$SAdd_{r4}$
10	1	25.8	19.1	413.2	86.6	2.5	1.5
10	1.5	36.2	25.0	451.1	89.0	1.9	1.4
10	2	30.1	24.6	515.2	115.2	2.1	1.4
15	1	29.5	27.7	435.0	89.3	1.5	1.3
15	1.5	37.0	35.4	610.4	138.9	3.6	0.9
15	2	35.1	36.1	779.7	163.6	3.3	1.0
20	1	27.4	31.5	525.4	115.3	2.2	0.9
20	1.5	39.1	40.7	775.9	168.7	4.0	1.1
20	2	52.8	43.4	1004.5	292.5	4.1	0.8

Table 5.11: Quantified variables for the $ED2$ design (1)

N	NS	minWT	MaxWT	minP	MaxP	$\overline{WT}(m)$	$\overline{P}$(€)	$\%\Delta WT$	$\%\Delta P$
$[8,12]$	$[1,1.4]$	0.3	26.3	210	1501	7.1	574.2	-26.6	38.3
$[8,12]$	$(1.4,1.8]$	0.1	31.0	305	1561	10.1	681.6	-29.3	32.5
$[8,12]$	$(1.8,2.2]$	3.2	36.5	396	1492	9.9	805.6	-36.8	20.0
$[13,17]$	$[1,1.4]$	3.1	51.6	364	1690	12.2	721.8	-31.2	29.7
$[13,17]$	$(1.4,1.8]$	5.8	59.3	419	1632	18.5	905.0	-27.7	21.8
$[13,17]$	$(1.8,2.2]$	5.1	63.2	538	1681	19.0	984.3	-29.4	16.4
$[18,22]$	$[1,1.4]$	6.0	63.1	515	1877	24.4	851.2	-32.4	21.7
$[18,22]$	$(1.4,1.8]$	7.3	60.7	638	1838	21.5	969.5	-36.8	16.1
$[18,22]$	$(1.8,2.2]$	8.7	69.2	801	1845	30.8	1252.4	-41.3	8.9

replanning in this 13.3% of cases. Then, $AQ4$ and consequently $MQ2$ are answered as true.

Comparing both $ED1$ and $ED2$ designs, it can be said that using both randomly generated data and real data from a log a similar behavior is observed in the bounds of the objective function values (cf. columns minWT, MaxWT, minP and MaxP in Tables 5.6 and 5.11). However, the upper bounds in the $ED2$ design (cf. columns MaxWT and MaxP in Tables 5.11) tend to be slightly different since the SDeclare model has been extended in this experiment (i.e., more services, more resources and uncertainty are included). The size of the ranges of the waiting time (i.e., differences between columns minWT and MaxWT) increases as the average number of services

Table 5.12: Quantified variables for the $ED2$ design (2)

N	NS	%Dominated	$\overline{Flex}$	minRob	MaxRob
$[8,12]$	$[1,1.4]$	100	100.0	22.2	44.4
$[8,12]$	$(1.4,1.8]$	100	100.0	16.7	38.9
$[8,12]$	$(1.8,2.2]$	100	95.6	22.2	27.8
$[13,17]$	$[1,1.4]$	100	100.0	25.0	41.7
$[13,17]$	$(1.4,1.8]$	100	95.6	16.7	33.3
$[13,17]$	$(1.8,2.2]$	100	95.6	16.7	20.8
$[18,22]$	$[1,1.4]$	100	88.9	16.7	30.0
$[18,22]$	$(1.4,1.8]$	100	95.6	13.3	23.3
$[18,22]$	$(1.8,2.2]$	100	83.3	13.3	16.7

per client increases (i.e., column NS). Such dependency is due to the fact that when NS is high, including additional services involves more complex schedules and therefore, it increases the waiting time more than when NS is low. However, the opposite happens with the size of the ranges of the profit. This is the expected behavior since the simpler the problems are (i.e., the lower values of NS), the more chances to include services exist. Therefore, the approach behaves similarly against both sources of problems.

3. In order to answer $MQ3$, the sub-questions $AQ5$, $AQ6$, $AQ7$, and $AQ8$ need to be answered. For this, Tables 5.13 is analyzed. The values of the columns $\#mQ$ and $\#MQ$ reveal that the number of questions that need to be answered for each node seems to be independent of the number of branches (cf. OB) of the related node. In addition, no errors were observed when generating the questionnaires and, thus, $AQ5$ can be answered as true. Furthermore, $\#MQ$ is lower than 10 (i.e., the number that the salon manager specified as maximum) and, thus, $AQ6$ can be answered as true.

   In order to answer $AQ7$ and $AQ8$, Table 5.14 is analyzed. As expected, $\#Q$ increases as $\#Acts$ increases, which indicates that more effort is required by the domain expert to individualize more complex configurable BP models. Even though, $\#Q$ is lower than 10 in all the cases, meaning that our approach can efficiently deal with real problems. Moreover, $\Delta\$$ increases as $\#Acts$ increases, which highlights the benefits of using the proposed approach in real cases and thus, $AQ7$ can be answered as true. Regarding the values of $\%R$, it can be concluded that the number of times that the salon manager needs to change her initial plan due to unexpected events (e.g., a client arrives later than expected or a resource becomes unavailable) is dras-

Table 5.13: Quantified variables for the $ED3$ design

OB	$\#mQ$	$\#MQ$
$[2,5)$	1.2	6.3
$[5,8)$	1.1	5.0
$[8,11)$	1.2	5.9
$[11,14)$	1.1	6.1

Table 5.14: Quantified variables for the $ED4$ design

$\#Acts$	$\#Q$	$\Delta\$$	$\%R$
$(40,60]$	3.1	141.5	70.0
$(60,80]$	4.1	189.1	60.0
$(80,100]$	7.6	219.4	66.7
$(100,120]$	8.0	239.8	75.0

tically reduced (i.e., almost 43% in most complex cases). Therefore, $AQ8$ and consequently $MQ3$ can be answered as true.

5.3.7 Validity Evaluation

This section evaluates if the results are valid and not biased. Three types of validity are addressed in this section: construct, internal and external.

Firstly, with relation to the construct validity, it has to be addressed in how far the measures which have been used are appropriate to address the research questions which have been planned. Three different threats are identified related to the acquisition of the data. The first threat is related to how the problems have been randomly generated in the $ED1$ design. In this design, unsolvable problems were not considered in order to evaluate the algorithm better. This is checked considering a simple rule: the generated appointment time of a client plus the time which her booked services consume cannot overpass the closing time of the beauty salon. Due to the parallelism which may exist because of the temporal constraints (i.e., a client can be served by different employees at the same time), this rule leaves out some problems which might be solvable. To mitigate this threat, a more elaborated algorithm can be performed to avoid eliminating problems which may be solvable. Secondly, the complexity of the problems which are generated is controlled only by varying the number of clients and her booked services (in the $ED1$ design), by the number of branches of the configurable nodes (in the $ED3$ design) and the number of activities of the configurable BP models (in the $ED4$ design). Although the beauty salon is considered a suitable business due to its complexity, different ways of controlling this complexity can be applied to mitigate this threat, e.g., by changing the type of constraints or by changing the properties specified by the salon manager. The third threat concerns the duration of the logged data in the designs (i.e, 90 days in case of $ED2$ design, and 30 days in case $ED3$ and $ED4$ designs). To the best of the acquired knowledge there is no metric which states how long data must be logged to obtain a meaningful log. To mitigate this threat, longer durations can be considered to get more data and

therefore to increase the probability of finding situations where the algorithm does not perform well. Finally, in all of the designs some values have been fixed in the algorithms, i.e., the number of regions into which the solution space is divided (fixed to 4) and the number of samples which are generated in the $ED2$ design (fixed to 30). Increasing these values would increase the accuracy of the proposal.

Regarding the internal validity, the main threat is that the obtained results related to flexibility and robustness concerns could be biased since their interpretation can be subjective since it depends on the business which is analyzed. In a similar way, the results concerning $\#MQ$ can be biased due to that the value for the upper bound of the number of questions to be answered to configure a specific configurable node (specified by the salon manager) represents a subjective point of view. This threat is difficult to eradicate. To mitigate it, other business experts can be consulted in order to state what is a successful value for flexibility and robustness and the upper bound of the number of questions per questionnaire.

Finally, the external validity considers in how far the obtained results could be generalized to any business. This generalization is threatened by the fact that the beauty salon was the unique scenario which was studied. Other scenarios can be considered to replicate this study in order to mitigate this threat.

Chapter 6

Discussion and Limitations

The manual specification of BP models, which are traditionally specified through an imperative language, can consume great quantity of resources, cause failures, and lead to non-optimized models, resulting in a very complex problem (Ferreira and Ferreira, 2006). The current approach allows modelling the considered problems in an easy way, since the considered declarative specifications (i.e., an extension of the Declare language (Pesic, 2008)) are based on high-level constraints. Moreover, with the proposed extension, the expressiveness of the process designs is enhanced compared to (Barba and Del Valle, 2011; Jimenez-Ramirez et al., 2013b,a) (e.g., stochastic values for modelling the uncertainty of the scenario can be included), and hence more realistic problems can be managed, e.g., the Beauty Salon detailed in Chapter. 5. Therefore, the current approach is intended to reduce the human work in scenarios with high variability in various ways:

- Since declarative BP model specifications allow their users to specify what has to be done instead of how (Pesic, 2008) and the tacit nature of human knowledge is often an obstacle to eliciting accurate process models (Ferreira and Ferreira, 2006), declarative specifications facilitate the human work which is involved in the process design and analysis phase compared to imperative specifications. Specifically, using a declarative specification, the user only has to define the constraints of her models without being aware of how they are fulfilled. Therefore, several ways of executing such declarative model exist. In turn, imperative specifications entail more complexity since all the possible execution alternatives need to be specified. Such complexity is even higher when a high flexibility is required, in the presence of input uncertainty, or when the resources need to be allocated in a suitable way considering the optimization of certain objective functions

- Typically, executing a declarative model (which presents high variability) usually entails bigger effort for the involved users compared to executing an

imperative model (Reichert and Weber, 2012; Schonenberg et al., 2008b) since deciding how to exactly execute the process is difficult for the user and this can lead to bad executions (i.e., very bad values for some objective functions). For this, the current approach extracts a desirable part of the variability of a declarative model through the generation of multi-objective optimized enactment plans while discarding bad execution alternatives (according to the optimization of some objective functions as well as high flexibility and robustness). This way, the proposed approach facilitates the human work which is involved in the process enactment phase.

- Such optimized plans are then merged into a configurable BP model which supports the analysts in the management of these plans and helps the analysts in understanding what the different plans share, what their differences are, and why and how these differences occur (Rosemann and van der Aalst, 2007).

Since these kind of problems are NP-complete, getting optimal solutions cannot be ensured in general (this is the reason why the term optimized plans is used instead of optimal plans). This way, the quality of the solution which is calculated depends on the time limit which is establish in the search algorithm. Note that, as mentioned, efficient filtering rules have been developed. Despite the NP-complexity of the considered problems, such filtering rules have demonstrated their effectiveness for improving performance in previous works (Barba and Del Valle, 2011; Barba et al., 2013a,b).

In addition, to further improve the quality of the resulting execution alternatives, flexibility and robustness concerns are considered. For this, quantitative definitions are provided in order to measure how the uncertainty of the scenario is supported by each generated enactment plan. Therefore, execution alternatives that are not desirable for the business regarding both the quality and a set of given objective functions are avoided.

Furthermore, in contrast to related proposals (Sadiq et al., 2005; Lu et al., 2009; Ferreira and Ferreira, 2006; Westergaard and Maggi, 2012; Krogt et al., 2010), not only a single enactment plan but a set of optimized enactment plans are considered when generating the imperative model. This way, the flexibility of the resulting imperative model is not unnecessarily restricted.

The optimized plans which are included in the generated configurable BP model can be used, as discussed in previous works, for:

1. Assisting users during the process execution to optimize performance through recommendations (Barba et al., 2011, 2013b).

2. Providing predictions, e.g., predict the completion time of all the running instances (Barba et al., 2013a).

3. Performing simulations, e.g., *what-if* scenarios (Barba et al., 2013a).

Moreover, the automatic generation of optimized configurable BP model can deal with complex and real problems in a simple way as demonstrated in Chapter 5. Therefore, a wide study of several aspects can be carried out by simulation. Nonetheless, an evaluation with more complex scenarios is required to improve the generalizability of the results and is planned for future work.

However, this approach also presents a few limitations. In general, different resource patterns can be taken into account. Motivated by the considered scenarios, the proposed approach considers that a resource can only perform an activity at the same time (i.e., the same resource cannot be used to perform more than one activity in parallel) and that activities are executed without preemption. The business analysts must deal with a new language for the constraint-based specification of BPs, thus a period of training is required to let them become familiar with the proposed language, i.e., SDeclare.[1] Furthermore, the optimized configurable BP model is generated by considering estimated values for the number of instances to execute, and hence the current proposal is only appropriate for processes in which this number is known a priori. As a real example, the beauty salon problem is detailed and an extensive empirical evaluation is carried out with the goal of supporting the contributions of the proposed approach. Some previous works also dealt with scenarios in which the number of process instances to be executed in a specific timeframe is known a priori (e.g., (Barba et al., 2013a) describes a travel agency problem and (Barba et al., 2012) considers computer support for clinical guidelines as an application example). In a related way, activity attributes and resource availability need to be estimated. Although this can be done more easily through the stochastic feature of SDeclare, the problem complexity increases as the number of stochastic variables increases. However, if the actual values deviate from the estimated values during the execution of the model, P&S techniques can be applied to replan the activities at runtime by considering the actual values of the estimates, as discussed in a previous work (Barba et al., 2013a).

In addition, motivated by the requirements of the considered scenarios, the data perspective which is considered in the current approach mainly includes data constraints which can be applied to input data and activity relations. However, more advanced features like dynamic data or data-flow perspective have been left out since they are not part of the design requirements of the considered scenarios and will be addressed in future work when applying the current proposal to BPs with different characteristics.

[1] To support the graphical specification of SDeclare models the existing Declare tool (available at http://www.win.tue.nl/declare/) has been adapted for allowing resource specification, temporal and data constraints, as well as stochastic estimates.

Chapter 7

Conclusions

Nowadays, there exists a growing interest in aligning information systems in a process-oriented way as well as in the effective and flexible management of business processes (BPs) since real scenarios are commonly subject to uncertainty. In order to manage such BPs, the traditional BP management life cycle involves four phases: (1) process design & analysis (i.e., a design of the BP is created following the requirements), (2) system configuration (i.e., the software defined in the BP design is implemented), (3) process enactment (i.e., the software is executed following the BP design) and (4) evaluation (i.e., monitoring information or logs are analyzed to look for design improvements).

In such context, analysts are in charge of designing BP models which capture the behavior of the business under analysis. To do this, such analysts must face certain design requirements, e.g., dealing with uncertainty estimates, resource allocation as well as control-flow, data and temporal constraints. In addition, the designed BP models typically have to optimize some objective functions –which may be opposed- while they must be able to withstand the input uncertainty to some extent (i.e., flexibility and robustness are required).

To address the aforementioned requirements, declarative BP models (e.g., constraint-based BP models) are increasingly used since the tacit nature of human knowledge is often an obstacle to eliciting accurate BP models. However, due to their flexible nature, there exist several ways of executing a declarative BP model, i.e., there are different imperative BP models associated to the same declarative specification. In such context, existing proposals (Pesic, 2008; Sadiq et al., 2005; Pesic et al., 2007; Lu et al., 2009; Ferreira and Ferreira, 2006; Montali, 2009; Westergaard and Maggi, 2012; Krogt et al., 2010; Hummer et al., 2013) generate a single execution plan from the declarative BP model. Nonetheless, if BPs are subject to uncertainty and conditions may change during BP execution, it might turn out that the generated enactment plan is not applicable and replanning might be required.

For overcoming this, the first part of this work (cf. Chapter 3) proposes the automatic generation of optimized enactment plans from constraint-based specifications at design-time. Such specifications consider the control-flow and resource perspectives as well as the specification of the input uncertainty of the scenario. For dealing with them, these plans are merged into a flexible configurable BP model in the second part of this work (cf. Chapter 4). Before the execution of such model, a single BP model has to be selected from it (i.e., it needs to be individualized). To individualize such models, unlike existing approaches (Rosa et al., 2009), a totally automated method to create a questionnaire-based application for guiding a business expert on individualizing the configurable BP model during run-time is proposed. This way the decision of how the enactment plan to be executed looks like is deferred to run-time.

To be more precise, the proposed approach includes the following contributions:

1. The definition of a suitable language, SDeclare, which allows the constraint-based specification of BPs be defined in a suitable form (both control-flow and resource perspectives are considered). Furthermore, such a language allows the analyst to include temporal and data constraints (cf. Chapter 3) as well as the input uncertainty which may exist (cf. Chapter 4). The proposed language significantly extends the existing proposals by considering multi-objective optimization, choice (Pesic, 2008), temporal (Montali, 2009; Westergaard and Maggi, 2012) and data constraints (Montali, 2009; Montali et al., 2013), alternative resources, and stochastic attributes.

2. A constraint-based proposal for planning and scheduling the BP activities in a multi-objective optimized way in order to obtain optimized BP enactment plans related to a SDeclare model (cf. Chapter 3). Unlike other related approaches (Pesic, 2008; Montali, 2009; Krogt et al., 2010; Lu et al., 2009; Rychkova et al., 2008; Hummer et al., 2013), the proposed method enables the optimization of several objective functions, resource reasoning and the specification of high level constraints.

3. A filtering algorithm which selects those plans which present an outstanding performance and which can deal with the highest uncertainty (cf. Chapter 4). Using such filtered plans, a configurable BP model is created out of them. For this, quantitative definitions of both flexibility and robustness are proposed unlike existing approaches (Cicerone et al., 2012; Aissi and Roy, 2010; Reichert and Weber, 2012; Schonenberg et al., 2008a).

4. Finally, a method for creating a questionnaire from the configurable BP model to allow the user individualize such model during run-time (cf. Chapter 4). Unlike (Rosa et al., 2009) which needs the intervention of an analyst

for creating the questionnaires, the current approach proposes a totally automatic method for generating such questionnaires.

In this way, once the analyst has designed the declarative BP model, the user is supported during the execution of it. Therefore, the loose of control which is commonly associated to the constraint-based specifications is ameliorated since the current approach enables the optimized execution of such flexible models.

The proposed approach is appropriate for managing scenarios which (1) present high variability, (2) pursue the optimization of some objective functions, (3) are subject to uncertainty, (4) have a well-defined set of business properties which can be extracted for a variant , and (5) highly rely on domain expert's skills (i.e., decisions influence business performance) and thus, decisions can not be predefined. Following this criteria, the beauty salon problem is selected (cf. Chapter 5) as case study. In order to validate the proposals of this Thesis Dissertation, a wide empirical evaluation is developed using such scenario. The results of such evaluation can be summarized as follows:

1. Although the optimization of enactment plans is a highly constrained problem, the proposed approach produces a satisfactory number of suitable solutions which, in addition, are well distributed in the solution space.

2. In most cases, the generated solutions help the user to improve the performance of her business.

3. In several cases, the current approach avoids making replanning due to the run-time feature. Therefore, selecting an execution plan incrementally during run-time produces a more suitable solution than selecting such plan before starting the execution.

In addition, the motivation and the interest related to the approach presented in the current document is strongly justified. Furthermore, discussions related to the advantages, drawbacks and limitations of each step of the approach are included. Moreover, the most related work together with the overcomings and innovations of the proposed approaches are also presented.

Chapter 8

Future Work

In this Thesis Dissertation a method for supporting the users during the execution of declarative business process models is presented. Such method includes a set of steps. Each one of these steps can be extended to widen the applicability of the proposed approach in different aspects, as explained as follows:

1. Related to the modelling language (i.e., SDeclare), the data perspective which is considered in the current approach mainly includes data constraints which can only be applied to input data and activity relations since it was motivated by the requirements of the considered scenarios (cf. Chapter 5). However, more advanced features like dynamic data or data-flow perspective have been left out since they are not part of the design requirements of the considered scenarios and will be addressed in future work when applying the current proposal to BPs with different characteristics. Furthermore, further resource patterns can be considered in the SDeclare language.

2. Regarding the algorithm which generates optimized enactment plans from SDeclare specifications, various additionally constraint-based solving techniques are planned to be explored in order to analyze their suitability for the generation of multi-objective optimized plans.

3. In the current approach the main application of the generated multi-objective optimized enactment plans (cf. Chapter 3) is the generation of configurable BP models to create the related questionnaires (cf. Chapter 4). Nonetheless, two additional scenarios are discusses in this Thesis Dissertation (i.e., user recommendations for optimized BP execution and the generation of optimized imperative BP models, cf. Section 3.3). However, other scenarios might be interesting to explore:

    - Performing simulations to investigate *what-if* scenarios or to generate a fast-forward view of the current process instance of a business pro-

cess model. For future work, the generated enactment plans can be used as a part of a simulation engine for declarativex models.

- Providing reliable time predictions (van der Aalst et al., 2011), e.g., to know when all the current process instances will terminate. Since all the temporal information is available in the generated enactment plans, these data can be used, as future work, for creating a prediction engine for declarative models.

4. An extended empirical study is intended to be carried out with the goal of measuring the impact of the robustness and flexibility on the process performance in different real businesses with more variety of activities, resources and sources of uncertainty, e.g., scenarios from information processing or manufacturing processes. In a related way, the tools associated to the proposed approach still need some refactoring in order to make it ready to use by others and thus, to make it publicly available.

5. Lastly, with relation to the automatic generation of questionnaires (cf. Chapter 4), some tasks are planned as future work:

   - Improve the semantics of the questions which are created since they seem too artificial in some cases. For this, the semantics of the business properties need to be improved. Therefore, a deeper research of the area of linguistics is required.
   - Analyze more in depth the different classification algorithms for creating the decision trees since the characteristics of them directly influence the questionnaire which is shown to the user.
   - Conduct experiments over other real scenarios for being able to generalize the results which are obtained.

Acronyms

AI Artificial Intelligence.

BP Busines Process.

BPM Busines Process Management.

BPMS Busines Process Management System.

COP Constraint Optimization Problem.

CP Constraint Programming.

CSP Constraint Satisfaction Problem.

MO-COP Multi-objective Constraint Optimization Problem.

P&S Planning and Schedulling.

RCPSP Resource-Constrained Project Scheduling Problem.

Appendices

Appendix A

SDeclare Basic Templates

As stated, in general, if not restricted by any constraints BP activities are assumed to be executed several times (Pesic and van der Aalst, 2006). Henceforth, $nt(A)$ refers to the number of times that the repeated activity A is executed; A_i represents the P&S activity related to the i-th execution of A; and $st(A_i)$ and $et(A_i)$ represent the start and the end times of A_i, respectively. It should be clarified that the constraints $\forall i : 1 \leq i < nt(A) : et(A_i) \leq st(A_{i+1})$ hold for each repeated activity A.

In Figure A.1, some representative examples of SDeclare templates are graphically represented. Specifically, three precedence relations between two repeated activities, A and B, are shown. As stated earlier, several executions of the same BP activity can be modelled as a sequence of single P&S activities. In this figure, the P&S activity A_i represents the i-th execution of the repeated activity A (A_i), and the arrow represents:

- A precedence relation between two P&S activities A_i and B_j, when it appears between two activities, which means that $et(A_i) \leq st(B_j)$.
- A precedence relation between a P&S activity A_i and a set S of P&S activities, when it appears between an activity and a dotted rectangle which encloses a set of activities, which means that $\exists B_j \in S : et(A_i) \leq st(B_j)$.
- A precedence relation between a set S of P&S activities and a P&S activity B_j, when it appears between a dotted rectangle which encloses a set of activities, and an activity, which means that $\exists A_i \in S : et(A_i) \leq st(B_j)$.

In a similar way, a special arrow (wider than the other arrows and with a big dot in its origin) which appears between two P&S activities, A and B, shows that A must be executed **immediately** before B ($et(A) = st(B)$). In a similar way, this can be defined for a set of activities. More details about Figure A.1 are shown in the definition of the related templates.

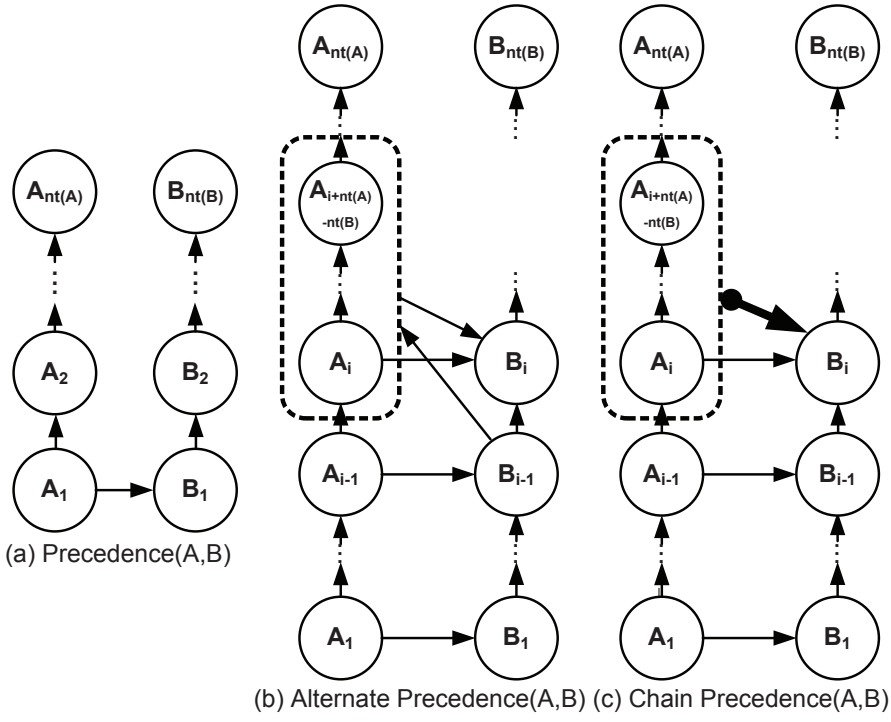

Figure A.1: Precedence templates when $nt(B) > 0$.

The SDeclare templates, based on Declare templates, together with some examples of valid and invalid traces[1], are listed as follows[2]. A full description of the Declare templates is included in the report (van der Aalst and Pesic, 2006b). These templates can be easily modified to include further possibilities.

- Existence(N,A): A must be executed more than or equal to N times, $nt(A) \geq N$.
- Absence(N,A): A must be executed less than N times, $nt(A) < N$.
- Exactly(N,A): A must be executed exactly N times, $nt(A) = N$.
- Responded Existence(A,B): If A is executed, then B must also be executed either before or after A, $nt(A) > 0 \Rightarrow nt(B) > 0$. For example, when Responded Existence(A,B) holds, <B>, <AB> or <BA> are valid traces, and <A> is an invalid trace since the execution of A requires the execution of B.

[1]For the sake of clarity, no parallelism between the activities is considered in the examples, i.e., trace $<A^1A^2\ldots A^n>$ means that $\forall i : 1 \leq i < n, et(A^i) = st(A^{i+1})$.

[2]For simplification, only non-branched templates are shown.

- CoExistence(A,B): The execution of A requires the execution of B, and vice versa, $nt(A) > 0 \iff nt(B) > 0$. For example, when CoExistence(A,B) holds, <AB> or <BA> are valid traces, and <B> is an invalid trace since the execution of B requires the execution of A.

- Precedence(A,B): Before the execution of B, A must have been executed, $nt(B) > 0 \Rightarrow (nt(A) > 0) \wedge (et(A_1) \leq st(B_1))$. As can be seen in Figure A.1(a), this relation implies that A_1 must precede B_1 in the case that $nt(B) > 0$. For example, when Precedence(A,B) holds, <ABBBA> is a valid trace, and <BAABB> is an invalid trace since the first B is executed before any A.

- Response(A,B): After the execution of A, B must be executed, $nt(A) > 0 \Rightarrow (nt(B) > 0) \wedge (st(B_{nt(B)}) \geq et(A_{nt(A)}))$. For example, when Response(A,B) holds, <BAABB> is a valid trace, and <ABBBA> is an invalid trace since after the last execution of A, B is not executed.

- Succession(A,B): Relations Precedence(A,B) and Response(A,B) hold. For example, when Succession(A,B) holds, <ABABB> is a valid trace, and <BABBA> is an invalid trace since the first B is executed before any A (moreover, after the last execution of A, B is not executed).

- 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. This implies that:

  1. The number of times that A is executed must be greater than or equal to the number of times that B is executed: $nt(A) \geq nt(B)$.
  2. Between each two executions of B, A must be executed at least once. Specifically, between the $(i-1)$-th and the i-th execution of B, the earliest execution of A that can exist is i, and hence A_{i-1} must precede B_{i-1} (as can be seen in Figure A.1(b)). In a similar way, between the $(i-1)$-th and the i-th execution of B, the latest execution of A that can exist is $i + nt(A) - nt(B)$, and hence B_i must precede $A_{i+nt(A)-nt(B)+1}$. This can also be seen in Figure A.1(b), where the possible activities to be executed between the $(i-1)$-th and the i-th execution of B are framed within the dotted rectangle. $\forall i : 2 \leq i \leq nt(B) : \exists j : i \leq j \leq i + nt(A) - nt(B) : st(A_j) \geq et(B_{i-1}) \wedge et(A_j) \leq st(B_i)$.
  3. Before the execution of B, A must be executed: $st(B_1) \geq et(A_1)$.

  For example, when Alternate Precedence(A,B) holds, <ABAABABA> is a valid trace, and <ABAABBAA> is an invalid trace since between the second and the third execution of B, there is not any A.

- Alternate Response(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. This implies:

  1. The number of times that B is executed must be greater than or equal to the number of times that A is executed: $nt(B) \geq nt(A)$.
  2. Between each two executions of A, B must be executed at least once. Specifically, between the i-th and the $(i+1)$-th execution of A, the earliest execution of B that can exist is i, and hence B_{i-1} must precede A_i. In a similar way, between the i-th and the $(i+1)$-th execution of B, the latest execution of A that can exist is $i+nt(B)-nt(A)-1$, and hence A_i must precede $B_{i+nt(B)-nt(A)}$. $\forall i : 1 \leq i < nt(A) : \exists j : i \leq j \leq i+nt(B)-nt(A)-1 : st(B_j) \geq et(A_i) \wedge et(B_j) \leq st(A_{i+1})$.
  3. After the execution of A, B must be executed: $st(B_{nt(B)}) \geq et(A_{nt(A)})$.

  For example, when Alternate Response(A,B) holds, <BABABBAB> is a valid trace, and <BAABBABB> is an invalid trace since between the first and the second execution of A, there is not any B.

- Alternate Succession(A,B): Both the relations AlternatePrecedence(A,B) and AlternateResponse(A,B) hold. For example, when Alternate Succession(A,B) holds, <ABABAB> is a valid trace, and <ABABBA> is an invalid trace since between the second and the third execution of B, there is not any A.

- Chain Precedence(A,B): **Immediately** before the execution of B, A must be executed. It implies that:

  1. The number of times that A is executed must be greater than or equal to the number of times that B is executed: $nt(A) \geq nt(B)$.
  2. Immediately before each execution of B, A must be executed. Specifically, before the i-th execution of B, the earliest execution of A that can exist is i. In a similar way, before the i-th execution of B, the latest execution of A that can exist is $i+nt(A)-nt(B)$. $\forall i : 1 \leq i \leq nt(B) : \exists j : i \leq j \leq i+nt(A)-nt(B) : et(A_j)$=$st(B_i)$.

  This is shown in Figure A.1(c), where a special arrow (wider than the other arrows and with a big dot in its origin) shows that A must be executed **immediately** before B. For example, when Chain Precedence(A,B) holds, <ABAABABA> is a valid trace, and <ABAABBAA> is an invalid trace since immediately before the third execution of B, there is not any A.

- Chain Response(A,B): **Immediately** after the execution of A, B must be executed. It implies:

  1. The number of times that B is executed must be greater than or equal to the number of times that A is executed: $nt(B) \geq nt(A)$.
  2. Immediately after each execution of A, B must be executed. Specifically, before the i-th execution of A, the earliest execution of B that can exist is i. In a similar way, after the i-th execution of A, the latest execution of B that can exist is $i + nt(B) - nt(A) - 1$. $\forall i : 1 \leq i \leq nt(A) : \exists j : i \leq j \leq i + nt(B) - nt(A) - 1 : st(B_j) = et(A_i)$.

  For example, when Chain Response(A,B) holds, <BABABBAB> is a valid trace, and <BAABBABB> is an invalid trace since immediately after the first execution of A, there is not any B.

- Chain Succession(A,B): Both the relations Chain Precedence(A,B) and Chain Response(A,B) hold. For example, when Chain Succession(A,B) holds, <ABABAB> is a valid trace, and <ABABBA> is an invalid trace since immediately before the third execution of B, there is not any A.

- Responded Absence and Not CoExistence(A,B): If B is executed, then A cannot be executed, and vice versa, $((nt(A) > 0) \cdot (nt(B) > 0)) == 0$. For example, when Responded Absence(A,B) holds, <A> or <B> are valid traces, and <BA> is an invalid trace.

- Negation Response, Negation Precedence, Negation Succession(A,B): After the execution of A, B cannot be executed, i.e., the last execution of B must finish before the start of the first execution of A ($nt(A) > 0 \wedge nt(B) > 0) \Rightarrow et(B_{nt(B)}) \leq st(A_1)$). For example, when Negation Succession(A,B) holds, <BBBA> is a valid trace, and <BBAB> is an invalid trace since the third B is executed after A.

- Negation Alternate Precedence(A,B): Between two executions of B, A cannot be executed, $nt(B) \geq 2 \Rightarrow \forall i : 1 \leq i \leq nt(A) : et(A_i) \leq st(B_1) \vee st(A_i) \geq et(B_nt(B))$. For example, when Negation Alternate Precedence(A,B) holds, <AABBA> is a valid trace, and <ABABA> is an invalid trace since between the first and the second execution of B, A is executed.

- Negation Alternate Response(A,B): Between two executions of A, B cannot be executed, $nt(A) \geq 2 \Rightarrow \forall 1 \leq i \leq nt(B) : et(B_i) \leq st(A_1) \vee st(B_i) \geq et(A_{nt(A)})$. For example, when Negation Alternate Response(A,B) holds, <BBAAB> is a valid trace, and <BABAB> is an invalid trace since between the first and the second execution of A, B is executed.

- Negation Alternate Succession(A,B): Both the relations Negation Alternate Precedence(A,B) and Negation Alternate Response(A,B) hold. For example, when Negation Alternate Succession(A,B) holds, <AABB> is a valid trace, and <AABBA> is an invalid trace since between the second and the third execution of A, B is executed.

- Negation Chain Succession(A,B): B cannot be executed immediately after the execution of A, $\forall i : 1 \leq i \leq nt(B) : \neg\exists j : 1 \leq j \leq nt(A) : et(A_j) = st(B_i)$. For example, when Negation Chain Succession(A,B) holds, <BACBA> is a valid trace, and <BABA> is an invalid trace since the second B is executed immediately after A.

The SDeclare templates can be classified either in unary (only one parameter, e.g., ExistenceN or AbsenceN) or binary (two parameters, e.g., Response or Chain Succession) templates.

Bibliography

Aaker, D., Mascarenhas, B., 1994. The need for strategic flexibility. California Management Review 5 (2), 72–89.

AbouRizk, S., Halpin, D., Wilson, J., 1994. Fitting beta distributions based on sample data. Journal of Construction Engineering and Management 120 (2), 288–305.

Aissi, H., Roy, B., 2010. Robustness in multi-criteria decision aiding. In: Ehrgott, M., Figueira, J. R., Greco, S. (Eds.), Trends in Multiple Criteria Decision Analysis. Vol. 142 of International Series in Operations Research & Management Science. pp. 87–121.

AristaFlow, 2009. AristaFlow BPM Suite BPM09-Demo. urlhttp://www.uni-ulm.de/einrichtungen/aristaflow-forum/screencasts.html, [Online; accessed 14-May-2014].

Athan, T. W., Papalambros, P. Y., 1996. A note on weigthed criteria methods for compromise solutions in multi-objective optimization. Engineering Optimization 27 (2), 155–176.

Back, W., Boles, W., Fry, G., 2000. Defining triangular probability distributions from historical cost data. Journal of Construction Engineering and Management 126 (1), 29–37.

Bao, F., Chintabathina, S., Morales, A., Rushton, N., Watson, R., Zhang, Y., 2011. A temporally expressive planner based on answer set programming with constraints: Preliminary design. In: Proc. LPNMR. pp. 398–414.

Barba, I., Del Valle, C., 2011. A Constraint-based Approach for Planning and Scheduling Repeated Activities. In: Proc. COPLAS. pp. 55–62.

Barba, I., Del Valle, C., Weber, B., Jimenez-Ramirez, A., 2013a. Automatic generation of optimized business process models from constraint-based specifications. Int J Cooper Inform Syst 22.

Barba, I., Lanz, A., Weber, B., Reichert, M., Del Valle, C., 2012. Optimized time management for declarative workflows. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (Eds.), Enterprise, Business-Process and Information Systems Modeling. Vol. 113 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 195–210.

Barba, I., Weber, B., Del Valle, C., 2011. Supporting the Optimized Execution of Business Processes through Recommendations. In: Proc. BPI.

Barba, I., Weber, B., Del Valle, C., Jimenez-Ramirez, A., 2013b. User recommendations for the optimized execution of business processes. Data & Knowledge Engineering 86 (0), 61 – 84.

Barták, R., Cepek, O., 2008. Incremental filtering algorithms for precedence and dependency constraints. International Journal on Artificial Intelligence Tools 17 (1), 205–221.

Bartak, R., Cepek, O., 2010. Incremental propagation rules for a precedence graph with optional activities and time windows. Trans. Inst. Meas. Control 32 (1), 73–96.

Beck, J., Fox, M., 1998. A generic framework for constraint-directed search and scheduling. AI Magazine 19 (4), 101 – 130.

Beck, J., Fox, M., 2000. Constraint-directed techniques for scheduling alternative activities. International Journal on Artificial Intelligence 121, 211–250.

BPEL, 2007. Web Services Business Process Execution Language Version 2.0: OASIS Standard. urlhttp://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, [Online; accessed 1-June-2011].

BPMN, 2011. Business Process Model and Notation (BPMN), Version 2.0. urlhttp://www.omg.org/spec/BPMN/2.0/, [Online; accessed 1-June-2011].

Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 41 (3), 157 – 183.

Breiman, L., 1984. Classification and regression trees. The Wadsworth and Brooks-Cole statistics-probability series. Chapman & Hall.

Brereton, P., Kitchenham, B., Budgen, D., 2008. Using a protocol template for case study planning. In: Proceedings of EASE 2008. BCS-eWiC.

Brown, K. N., Miguel, I., 2006. Uncertainty and change. In: Handbook of Constraint Programming, chapter 21. Elsevier.

Brucker, P., Heitmann, S., Hurink, J., Nieberg, T., 2006. Job-shop scheduling with limited capacity buffers. OR Spectrum 28 (2), 151–176.

Brucker, P., Knust, S., 2006. Complex Scheduling (GOR-Publications). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Carlson, J. M., Doyle, J., 2002. Complexity and robustness. Proceedings of the National Academy of Sciences of the United States of America 99 (Suppl 1), 2538–2545.

Chankong, V., Haimes, Y., 1983. Multiobjective Decision Making Theory and Methodology. Elsevier.

Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S., 2009. Exploiting inductive logic programming techniques for declarative process mining. In: Transactions on Petri Nets and Other Models of Concurrency II. Springer, pp. 278–295.

Cicerone, S., Di Stefano, G., Schachtebeck, M., Schöbel, A., May 2012. Multi-stage recovery robustness for optimization problems: A new concept for planning under disturbances. Inf. Sci. 190, 107–126.

Das, I., Dennis, J. E., 1997. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Structural and Multidisciplinary Optimization 14, 63–69.

Davenport, T. H., 1993. Process innovation: reengineering work through information technology. Harvard Business School Press.

de Haan, J., Kwakkel, J., Walker, W., Spirco, J., Thissen, W., 2011. Framing flexibility: Theorising and data mining to develop a useful definition of flexibility and related concepts. Futures 43 (9), 923 – 933, special Issue: Flexible infrastructures.

Deb, K., Kalyanmoy, D., 2001. Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc., New York, NY, USA.

Dechter, R., Dechter, A., 1988. Belief maintenance in dynamic constraint networks. pp. 37–42.

Declare, 2011. Declare: Declarative Approach to Workflow Management Systems. http://www.win.tue.nl/declare/, [Online; accessed 1-May-2014].

Demeyer, R., Van Assche, M., Langevine, L., Vanhoof, W., 2010. Declarative workflows to efficiently manage flexible and advanced business processes. In: Proc. PPDP. pp. 209–218.

der Aalst, W., Dreiling, A., Gottschalk, F., Rosemann, M., Jansen-Vullers, M., 2006. Configurable process models as a basis for reference modeling. In: Bussler, C., Haller, A. (Eds.), Business Process Management Workshops. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 512–518.

Dijkman, R., Rosa, M. L., Reijers, H. A., Feb. 2012. Managing large collections of business process models-current techniques and challenges. Comput. Ind. 63 (2), 91–97.

Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., Zbyslaw, A., 1996. Freeflow: Mediating between representation and action in workflow systems. In: Proc. CSCW. pp. 190–198.

Drabble, B., Tate, A., 1994. The use of optimistic and pessimistic resource profiles to inform search in an activity based planner. In: Proc. AIPS. pp. 243–248.

Drexl, A., Gruenewald, J., 1993. Nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions (Institute of Industrial Engineers) 25 (5), 74–81.

Dubois, H. F., Prade, H., 1993. Using fuzzy constraints in job-shop scheduling. In: In Proc. of IJCAI93/SIGMAN Workshop on Knowledge-based Production Planning,Scheduling and Control.

Ehrgott, M., Gandibleux, X., 2003. Multiobjective combinatorial optimization - theory, methodology, and applications. In: Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys. Vol. 52 of International Series in Operations Research & Management Science. pp. 369–444.

Eppink, D. J., 1978. Planning for strategic flexibility. Long Range Planning 11 (4), 9 – 15.

Erol, K., Hendler, J., Nau, D., 1994. HTN planning: Complexity and expressivity. In: Proc. of 20th AAAI Conference. pp. 1123–1128.

Faltings, B., Macho-Gonzalez, S., 2005. Open constraint programming. Artificial Intelligence 161 (12), 181 – 208, distributed Constraint Satisfaction.

Fargier, H., Lang, J., 1993. Uncertainty in constraint satisfaction problems: A probabilistic approach. pp. 97–104.

Fargier, H., Lang, J., Schiex, T., 1996. Mixed constraint satisfaction: a framework for decision problems under incomplete knowledge. pp. 175–180.

Fente, J., Schexnayder, C., Knutson, K., 2000. Defining a probability distribution function for construction simulation. Journal of Construction Engineering and Management 126 (3), 234–241.

Feo, T., Resende, M., 1989. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8, 67–71.

Feo, T., Resende, M., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133.

Fernandes, R., Lange, F., Burchett, R., Happ, H., Wirgau, K., 1983. Large scale reactive power planning. IEEE transactions on power apparatus and systems PAS-102 (5), 1083–1088.

Ferreira, H., Ferreira, D., 2006. An integrated life cycle for workflow management based on learning and planning. Int J Cooper Inform Syst 15 (4), 485 – 505.

Fikes, R., Nilsson, N., 1971. Strips: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2, 189–208.

Fowler, D., Brown, K., 2000. Branching constraint satisfaction problems for solutions robust under likely changes.

Gabriel, G., Grandcolas, S., 2009. Searching optimal parallel plans: A filtering and decomposition approach. pp. 576–580.

Gantt, H., 1913. Work, wages, and profits. Engineering Magazine Co.

Garey, M. R., Johnson, D. S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman & Co.

Garrido, A., Arangu, M., Onaindia, E., 2009. A constraint programming formulation for planning: From plan scheduling to plan generation. Journal of Scheduling 12 (3), 227–256.

Garrido, A., Onaindia, E., Sapena, O., 2008. Planning and scheduling in an e-learning environment. a constraint-programming-based approach. Engineering Applications of Artificial Intelligence 21 (5), 733–743.

Georgakopoulos, D., Hornick, M., Sheth, A., 1995. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases 3, 119–153.

Ghallab, M., Nau, D., Traverso, P., 2004. Automated Planning: Theory and Practice. Morgan Kaufmann, Amsterdam.

Glover, F., 1989. Tabu search part i. Orsa Journal on Computing 1 (3), 190–206.

Golden, W., Powell, P., 2000. Towards a definition of flexibility: in search of the holy grail? Omega 28 (4), 373 – 384.

Gomes, C., Van Hoeve, W., Selman, B., 2006. Constraint programming for distributed planning and scheduling. Vol. SS-06-04. pp. 157–158.

Gottschalk, F., Jansen-vullers, M. H., 2006. Configurable process models a foundational approach. In: In Proceedings of the Reference Modelling Conference 2006. Springer, pp. 51–66.

Gottschalk, F., van der Aalst, W. M., Jansen-Vullers, M. H., La Rosa, M., 2008. Configurable workflow models. International Journal of Cooperative Information Systems (IJCIS) 17 (2), 177–221.

Group, G., 2010. Leading in Times of Transition: The 2010 CIO Agenda (EXP Premier Report No. January 2010). In: Report, Gartner, Inc.

Gueorguiev, S., Harman, M., Antoniol, G., 2009. Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. GECCO '09. New York, NY, USA, pp. 1673–1680.

Haimes, Y., Lasdon, L., Wismer, D., 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Trans on Syst Man Cybern 1, 296–297.

Hallerbach, A., Bauer, T., Reichert, M., 2010. Capturing variability in business process models: The provop approach. Journal of Software Maintenance and Evolution: Research and Practice 22 (6-7), 519–546.

Hammond, K., 1990. Case-based planning: A framework for planning from experience. Cognitive Science 14 (3), 385–443.

Herroelen, W., Leus, R., Sep. 2005. Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research 165 (2), 289–306.

Hummer, W., Gaubatz, P., Strembeck, M., Zdun, U., Dustdar, S., 2013. Enforcement of entailment constraints in distributed service-based business processes. Information and Software Technology 55 (11), 1884 – 1903.

Jensen, M., 2001. Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. Journal of Applied Soft Computing 1, 35–52.

Jensen, M., 2003. Generating robust and flexible job shop schedules using genetic algorithms. Evolutionary Computation, IEEE Transactions on 7 (3), 275–288.

Jimenez-Ramirez, A., Barba, I., Del Valle, C., Weber, B., 2013a. Generating multi-objective optimized business process enactment plans. In: Salinesi, C., Norrie, M., Pastor, s. (Eds.), Advanced Information Systems Engineering. Vol. 7908 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 99–115.

Jimenez-Ramirez, A., Barba, I., Del Valle, C., Weber, B., 2013b. Optbpplanner: Automatic generation of optimized business process enactment plans. In: Linger, H., Fisher, J., Barnden, A., Barry, C., Lang, M., Schneider, C. (Eds.), Building Sustainable Information Systems. Springer US, pp. 429–442.

Joshi, S., Kersting, K., Khardon, R., 2011. Decision-theoretic planning with generalized first-order decision diagrams. Artificial Intelligence 175 (18), 2198–2222.

Karim, A., Arif-Uz-Zaman, K., 2013. A methodology for effective implementation of lean strategies and its performance evaluation in manufacturing organizations. Business Process Management Journal 19 (1), 169–196.

Kirkpatrick, S., Gelatt, C., Vecchi, M., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.

Koehler, J., 1998. Planning under resource constraints. In: Proc. ECAI. pp. 489–493.

Koski, J., 1985. Defectiveness of weighting method in multicriterion optimization of structures. Comm Numer Meth Eng 1 (6), 333–337.

Krogt, R., Geraghty, J., Salman, M. R., Little, J., 2010. On supporting lean methodologies using constraint-based scheduling. J. of Scheduling 13, 301–314.

Kumar, A., Yao, W., 2012. Design and management of flexible process variants using templates and rules. Computers in Industry 63 (2), 112 – 130, managing Large Collections of Business Process Models Managing Large Collections of Business Process Models.

L. Climent, R. J. Wallace, M. A. S., Barber, F., 2014. Robustness and stability in constraint programming under dynamism and uncertainty. Journal of Artificial Intelligence Research 49, 49–78.

La Rosa, M., Dumas, M., ter Hofstede, A. H., 2008. Modelling business process variability for design-time configuration. Handbook of Research on Business Process Modeling.

Laborie, P., Ghallab, M., 1995. Planning with sharable resource constraints. In: Proc. IJCAI. pp. 1643–1649.

Laborie, P., Rogerie, J., Shaw, P., Vilim, P., 2009. Reasoning with conditional time-intervals. part ii: An algebraical model for resources.

Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S., 2007. Inducing declarative logic-based models from labeled traces. In: Business Process Management. Springer, pp. 344–359.

Laumanns, M., T. L. Z. E., 2006. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. European Journal of Operational Research 169 (3), 932–942.

Leitmann, G., 1977. Some problems of scalar and vector-valued optimization in linear viscoelasticity. IEEE Trans on Automat Control 23, 93–99.

Lekavy, M., Návrat, P., 2007. Expressivity of STRIPS-Like and HTN-Like Planning. In: Proc. KES-AMSTA. pp. 121–130.

Liu, Y., Jiang, Y., 2006. Lp-tpop: Integrating planning and scheduling through constraint programming. In: Proc. PRICAI. pp. 844–848.

Lombardi, M., Milano, M., 2010. Constraint based scheduling to deal with uncertain durations and self-timed execution. In: Proc. CP. pp. 383–397.

Lu, R., Sadiq, S., Governatori, G., 2009. On managing business processes variants. Data Knowl. Eng. 68 (7), 642 – 664.

Lu, R., Sadiq, S., Padmanabhan, V., Governatori, G., 2006. Using a temporal constraint network for business process execution. In: Proc. ADC. pp. 157–166.

Ly, L., Rinderle, S., P., D., 2008. Integration and verification of semantic constraints in adaptive process management systems. Data Knowl Eng 64 (1), 3 – 23.

Maio, C., Schexnayder, C., Knutson, K., Weber, S., 2000. Probability distribution functions for construction simulation. Journal of Construction Engineering and Management 126 (4), 285–292.

Mendling, J., Neumann, G., van der Aalst, W. M. P., 2007. Understanding the occurrence of errors in process models based on metrics. Lecture Notes in Computer Science LNCS 4803, 113–130.

Messac, A., Puemi-Sukam, C., Melachrinoudis, E., 2000. Aggregate objective functions and pareto frontiers: Required relationships and practical implications. Optimization and Engineering 1, 171–188.

Mitchell, M., 1998. An Introduction to Genetic Algorithms. MIT Press.

Monette, J., Deville, Y., Van Hentenryck, P., 2009. Just-in-time scheduling with constraint programming. pp. 241–248.

Montali, M., 2009. Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach. Ph.D. thesis, Department of Electronics, Computer Science and Telecommunications Engineering, University of Bologna.

Montali, M., Chesani, F., Mello, P., Maggi, F. M., 2013. Towards data-aware constraints in declare. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. SAC '13. pp. 1391–1396.

Montali, M., Pesic, M., Aalst, W. M. P. v. d., Chesani, F., Mello, P., Storari, S., 2010. Declarative specification and verification of service choreographiess. ACM Trans. Web 4 (1), 3:1–3:62.

Mouhoub, M., Sadaoui, S., Sukpan, A., 2003. Chronological backtracking versus formal methods for solving CSPs. In: Proc. of the International Conference on Artificial Intelligence IC-AI 2003. pp. 270–275.

Moura, A., De Souza, C., Cire, A., Lopes, T., 2008. Heuristics and constraint programming hybridizations for a real pipeline planning and scheduling problem. pp. 455–462.

Nareyek, A., Freuder, E., Fourer, R., Giunchiglia, E., Goldman, R., Kautz, H., Rintanen, J., Tate, A., 2005. Constraints and ai planning. IEEE Intelligent Systems 20 (2), 62 – 72.

Nuijten, W., Aarts, E., 1996a. A computational study of constraint satisfaction for multiple capacitated job shop scheduling. European Journal of Operational Research 90 (2), 269 – 284.

Nuijten, W., Aarts, E., 1996b. Sequencing with earliness and tardiness penalties: a review. European Journal of Operational Research 90 (2), 269 – 284.

Ouyang, C., van der Aalst, W. M., Dumas, M., ter Hofstede, A. H., October 2006. From business process models to process-oriented software systems: The bpmn to bpel way.

Penberthy, J., Weld, D., 1994. Temporal planning with continuous change. In: Proc. AAAI. pp. 1010–1015.

Pesic, M., 2008. Constraint-Based Workflow Management Systems: Shifting Control to Users. Ph.D. thesis, Eindhoven University of Technology, Eindhoven.

Pesic, M., Schonenberg, M., Sidorova, N., van der Aalst, W., 2007. Constraint-Based Workflow Models: Change Made Easy. In: OTM Conferences (1). pp. 77–94.

Pesic, M., van der Aalst, W., 2006. A declarative approach for flexible business processes management. In: Proc. BPM Workshops. pp. 169–180.

Pinedo, M. L., 2008. Scheduling: Theory, Algorithms, and Systems, 3rd Edition. Springer Publishing Company, Incorporated.

Pisinger, D., Ropke, S., 2010. Lar ge neighborhood search. International Series in Operations Research & Management Science 146 (13), 399–420.

PSL, 1977. Process Specification Language project. urlhttp://www.nist.gov/psl/, [Online; accessed 1-May-2012].

Reichert, M., Weber, B., 2012. Enabling Flexibility in Process-Aware Information Systems. Springer Berlin Heidelberg.

Reijers, H., 2003. Design and Control of Workflow Processes. Springer-Verlag Berlin, Heidelberg.

Rosa, M., Aalst, W., Dumas, M., ter Hofstede, A., 2009. Questionnaire-based variability modeling for system configuration. Software & Systems Modeling 8 (2), 251–274.

Rosa, M. L., Dumas, M., Kaarik, R., Dijkman, R. M., 2010. Merging business process models. In: 18th International Conference on Cooperative Information Systems (CoopIS '10) 2010. Springer-Verlag Heidelberg, Crete, Greece, pp. 96–113.

Rosa, M. L., Dumas, M., ter Hofstede, A. H., Mendling, J., 2011. Configurable multi-perspective business process models. Information Systems 36 (2), 313 – 340, special Issue: Semantic Integration of Data, Multimedia, and Services.

Rosa, M. L., Dumas, M., Uba, R., Dijkman, R. M., 2012. Business process model merging : An approach to business process consolidation. ACM Transactions on Software Engineering and Methodology (TOSEM).

Rosemann, M., van der Aalst, W., 2007. A configurable reference modelling language. Inform Syst 32, 1–23.

Rossi, F., van Beek, P., Walsh, T. (Eds.), 2006. Handbook of Constraint Programming. Elsevier.

Russell, N., van der Aalst, W., ter Hofstede, A., Edmond, D., 2005. Workflow resource patterns: Identification, representation and tool support. In: Proc. CAiSE. pp. 216–232.

Rychkova, I., Regev, G., Wegmann, A., 2008. Using declarative specifications in business process design. IJCSA 5 (3b), 45–68.

Sabin, M., Freuder, E., 1998. Detecting and resolving inconsistency and redundancy in conditional constraint satisfaction problems. In: Proceeding of Constraint Programming (CP'98). pp. 90–94.

Sadiq, S. W., Orlowska, M. E., Sadiq, W., 2005. Specification and validation of process constraints for flexible workflows. Inform Syst 30 (5), 349–378.

Salido, M., 2010. Introduction to planning, scheduling and constraint satisfaction. J Intell Manuf 21 (1), 1–4.

Schnieders, A., Puhlmann, F., 2006. Variability mechanisms in e-business process families. In: Proc. International Conference on Business Information Systems (BIS 2006. pp. 583–601.

Schonenberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W., 2008a. Process flexibility: A survey of contemporary approaches. In: Advances in Enterprise Engineering I. Springer, pp. 16–30.

Schonenberg, H., Weber, B., Dongen, B., Aalst, W., 2008b. Supporting flexible processes through recommendations based on history. In: Dumas, M., Reichert, M., Shan, M.-C. (Eds.), Business Process Management. Vol. 5240 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 51–66.

Schunselaar, D., Maggi, F., Sidorova, N., Aalst, W., 2012. Configurable declare: Designing customisable flexible process models. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I. (Eds.), On the Move to Meaningful Internet Systems: OTM 2012. Vol. 7565 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 20–37.

Shah, R., Ward, P. T., 2003. Lean manufacturing: context, practice bundles, and performance. Journal of Operations Management 21 (2), 129 – 149.

Shukla, P. K., Deb, K., 2007. On finding multiple pareto-optimal solutions using classical and evolutionary generating methods. European Journal of Operational Research 181 (3), 1630 – 1652.

Smith, D., Frank, J., Jónsson, A., 2000. Bridging the gap between planning and scheduling. Knowledge Engineering Review 15(1), 47–83.

Smith, D., Weld, D., 1999. Temporal planning with mutual exclusion reasoning. In: Proc. IJCAI. pp. 139–144.

Smith, K., Everson, R., Fieldsend, J., Murphy, C., Misra, R., June 2008. Dominance-based multiobjective simulated annealing. Evolutionary Computation, IEEE Transactions on 12 (3), 323–342.

Souki, M., 2011. Operating theatre scheduling with fuzzy durations. J. Appl. Oper. Res. 3, 177–191.

Stadler, W., 1995. Caveats and boons of multicriteria optimization. Computer-Aided Civil and Infrastructure Engineering 10 (4), 291–299.

Stevenson, M., Spring, M., 2007. Flexibility from a supply chain perspective: definition and review. International Journal of Operations & Production Management 27 (7), 685–713.

Suman, B., 2004. Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. Computers & Chemical Engineering 28 (9), 1849 – 1871.

Sun, C., Aiello, M., 2008. Towards variable service compositions using vxbpel. In: Proceedings of the 10th international conference on Software Reuse: High Confidence Software Reuse in Large Systems. ICSR '08. pp. 257–261.

Timpe, C., 2002. Solving planning and scheduling problems with combined integer and constraint programming. OR Spectrum 24 (4), 431–448.

Tu, P., Son, T., Pontelli, E., 2007. CPP: A constraint logic programming based planner with preferences. In: Proc. LPNMR. pp. 290–296.

Upton, D., 1994. The management of manufacturing flexibility. California Management Review 36 (2), 72–89.

van der Aalst, W., Pesic, M., 2006a. DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Technical Report BPM-06-21, BPMcenter.org.

van der Aalst, W., Pesic, M., 2006b. Specifying, discovering, and monitoring service flows: Making web services process-aware. In: Technical Report BPM-06-09, BPMcenter.org.

van der Aalst, W., Pesic, M., Schonenberg, H., 2009. Declarative workflows: Balancing between flexibility and support. Computer Science - Research and Development 23 (2), 99–113.

van der Aalst, W., Schonenberg, M., Song, M., 2011. Time prediction based on process mining. Inform Syst 36 (2), 450–475.

van der Aalst, W., ter Hofstede, A., Weske, M., 2003. Business Process Management: A Survey. In: Proc. BPM. pp. 1–12.

van Dongen, B., 2007. Process Mining and Verification. Ph.D. thesis, Eindhoven University of Technology, Eindhoven.

Vanderfeesten, I., Reijers, H., van der Aalst, W., 2008. Product based workflow support: A recommendation service for dynamic workflow execution. In: Technical Report BPM-08-03, BPMcenter.org.

Verfaillie, G., Jussien, N., 2005. Constraint solving in uncertain and dynamic environments: A survey. Constraints 10 (3), 253–281.

Vidal, V., Geffner, H., 2006. Branching and pruning: An optimal temporal pocl planner based on constraint programming. Artificial Intelligence 170 (3), 298–335.

Vossen, T., Ball, M., Lotem, A., Nau, D., 1999. On the use of integer programming models in ai planning. The Knowledge Engineering Review 15 (1).

Wainer, J., De Lima Bezerra, F., 2003. Constraint-based flexible workflows. In: Proc. CRIWG. pp. 151–158.

Walsh, T., 2002. Stochastic constraint programming.

Weske, M., 2007. Business Process Management: Concepts, Languages, Architectures. Springer.

Westergaard, M., Maggi, F., 2012. Looking into the future: Using timed automata to provide a priori advice about timed declarative process models. In: International Conference on Cooperative Information Systems (CoopIS 2012).

Wolfman, S., Weld, D., 1999. Combining linear programming and satisfiability solving for resource planning. The Knowledge Engineering Review 15 (1).

Zeleny, M., 1982. Multiple Criteria Decision Making. McGraw-Hill, New York.

Zilberstein, S., 1996. Using anytime algorithms in intelligent systems. AI Magazine 17 (3), 73–83.

Zugal, S., Soffer, P., Pinggera, J., Weber, B., 2012. Expressiveness and Understandability Considerations of Hierarchy in Declarative Business Process Models. In: Proc. BPMDS '12. pp. 167–181.