New Trends in Software Methodologies, Tools and Techniques
H. Fujita and R. Revetria (Eds.)
IOS Press, 2012

doi:10.3233/978-1-61499-125-0-215

Enhancing Business Process Management with a Constraint-Based Approach

Wolfgang RUNTE
Institute of Computer Science
University of Osnabrueck
D-49069 Osnabrueck, Germany
woru@informatik.uni-osnabrueck.de

Abstract. In the 1980s rule-based systems became very popular in the domain of expert systems. It soon became apparent, that large rule-based systems causes enormous maintenance problems, because of the lack of separation between domain knowledge an control strategy. Rules are declarative, weakly structured, difficult to manage and maintain and should be applied only in local contexts and with limited use. For large rule bases the user can not be sure if the problem is completely covered by the rules, modifications often result in unwanted consequences. For application in business process management (BPM) rules became popular again, but the known insufficiencies still remain. Today it seems to be quite ignored that techniques using rules have strucural drawbacks that limit their application significantly. We present a constraint-based approach to enhance process models with additional knowledge. Constraints allow compact modeling of decision processes to inference specific values in process models. Furthermore constraints may be used as mechanism for quality assurance (QA). Constraints also have a declarative paradigma but avoid the lack of separation between domain knowledge and control strategy. Constraint solvers used as black box by a process engine will merely compute an output based on the given domain knowledge and return these as new input for the process engine. The constraint solver will give control as soon as possible back to the process engine. The process engine exclusively has to decide how to proceed. So constraints support a process engine without competing for the control strategy.

Keywords. business process modeling, business rules, maintenance problem, constraint satisfaction

1. Motivation

In business process management (BPM) compilances and relations between processes have to be considered by process engines. For the definition of relations and for compliance management mainly the rule-based approach, namely *business rules*, is applied [1]. The control flow in BPM systems is supported by rule engines. Decisions are taken on basis of dependencies and relations between entities defined in business rules. A practical field of application of business rules is

the area of ERP systems (*enterprise resource planning*) [2], [3], [4], [5]. Although there was almost a hype around the application of business rules in the last years there exist a number of well known drawbacks in the usage of rules in software systems. Today it is quite disregarded that already in the 1980s the excessive usage of rules showed evident insufficiencies.

1.1. Drawbacks of Rule-Based Systems

In the early days of expert systems (later on called knowledge-based systems) the research and development were dominated by the rule-based approach, so expert systems were characterised as *rule-based systems*. Knowledge-based systems for synthesis tasks like product configuration used rules as inference mechanism for the first time in increased dimensions [6,7]. It soon became apparent that large rule-based systems causes enormous maintenance problems [8,9,10]. Rules specify *directed relationships* as well as *actions*. While a directed relationship represents domain knowledge between entities, an action represents procedural knowledge to control the execution of a task. The lack of separation between domain knowledge and control strategy and also the sharing of knowledge about a single entity over several rules make the knowledge maintenance an extremly difficult task [11,12].

Consider the problem of updating rules with respect to changes in specific entities. It is very hard and very costly to be certain that all rules were found that need to be updated. Moreover the rule engine does not execute the action part if the condition of a rule is not up to date. So every maintenance task has to be performed very carefully and accurately what will result in high costs for each update. Otherwise one cannot be sure a specific rule is considered at all. The developer of the rule base has to ensure that all required conditions are covered by the set of rules and all desired actions can be reached. So also small changes will generate much costs [12].

To get an idea about the complexity of the maintenance task in rule-based systems there exists the well documented example of the rule-based configuration system R1/XCON, a well known knowledge-system for configuring VAX computer systems at Digital Equipment Corporation (DEC) [13,6]. R1/XCON used OPS-5, a production rule programming language. In 1989 XCON had in its rule base more than 31,000 object descriptions and approximately 17,500 rules [10]. Of these rules approximately 40-50 percent have to be modified or regenerated every year. The maintenance of the knowledge base was carried out by up to 40 employees [14]. Often the developers weren't sure about the intended purpose of a certain rule, so modifications resulted in unwanted consequences [15]. Because of the extreme maintenance costs some authors claimed that XCON was no longer maintainable [16].

This challenge leads to modularization and the subsumption of rules to *meta rules*. A programming methodology called RIME was developed that provides structuring concepts for rule-based programming [9,17]. Meta rules are used to control and order context specific decisions. The developer is able to force the firing of rules in a fixed sequence and therefore decomposing the problem into steps. When entering a step, the system activates the satisfied rules and the process proceeds to the next step. Indirectly, this provides some guidance in organizing

the rule base, but because of the huge number of rules, the maintenance problem remains still unsolved [12]. Furthemore numerousness additional dependencies of the rules between each other had to be managed, and rules were no independent knowledge units any more [16].

R1/XCON is a good illustration that lead to the understanding that rules are only weakly structured and should only be used in manageable domains in a straightforward manner. Because of the known insufficiencies rules should be applied only locally and with limited use. The rule-based approach is not well suited for the global control of processes at all [16]. With respect to process engines this means rules are suited only for limited application. Rules are well suited to represent local, causal relationships (implications), but the global control should be performed exclusively by the process engine on basis of the process model.

Notwithstanding the process engine may be combined with other techniques. Instead of rules dependencies and relations can be modeled with *constraints* to eliminate the risk of mixing domain knowledge and control stragegy. A *constraint solver* will support the process engine using *constraint satisfaction* techniques to ensure the consistency of the modeled dependencies. Here the *black box* principle guarantees that the constraint solver is not used to take control over the prozess engine.

1.2. Example: Difference between Rules and Constraints

A simple example will illustrate the difference between a rule and a constraint. Imagine the process model in the scenario of the organization and realization of a holiday tent camping of a children's group. The following is an example of an important rule during this camping:

```
If there is a storm warning, then
strike the tents and evacuate the camp.
```

Notice that the rule contains some control knowledge, i. e. which activities have to be executed next if the condition is fulfilled. In the same scenario the appropriate constraint would be simple like this:

```
no storm warning
```

Or in a slightly more formally way:

```
stormWarning == false
```

Which means that there must not be a storm warning, otherwise the constraint is not satisfied. A constraint violation would mean the process model is in an inconsistent state, for which e. g. a sort of exception handling would be an appropriate reaction by the process engine. The difference to a rule-based approach is that the control knowledge, defining what to do if the constraint does not hold (*strike the tents and evacuate the camp*) has to be modeled as a process model executed by the process engine, *not* by a rule engine.

Constraints (and so rules) should contain as less as possible control knowledge. Instead the process engine should get back the flow control as soon as possi-

ble. Notice that in the constraint above there is no information about what to do if the constraint is violated. Ergo there is no control knowledge in this constraint, while in the rule concrete actions are specified. What to do if the constraint is violated, that means the variable `stormWarning` would have the value `true`, has to be specified elsewhere. For this a separate component is needed that decides how to proceed in the case of a constraint violation.

The above example is a simple *unary* constraint containing only one variable: `stormWarning`. More constraints with more variables would form a net of constraints and the change of the value of a variable can be propagated throughout this constraint net with constraint satisfaction techniques. So with constraint propagation it is possible to generate inferences on the basis of the knowledge represented by constraints.

In this paper we will show that constraint satisfaction is an adequate technique to model and manage dependencies in business processes. It is an alternative for the usage of rule-based approaches, which have well known insufficiencies. Because of the domain indendency and the absence of control strategy constraints may become an important role, e. g. in the field of quality assurance (QA) of business processes.

In the following section 2 the principles of the rule-based approach and examples of current rule engines are discussed. In section 3 we give an overview over the principles of the constraint-based approach and the application of constraints in business process management. In section 4 examples for constraints in business processes models are shown and furthermore a classification in different classes of constraints is elaborated. In section 5 a basic discussion with respect to benefits of the constraint mechanism is given. Section 6 contains related work and in section 7 a conclusion and a short outlook is given.

2. Rule-Based Systems

The usage of rules in the development of software systems has a long history. In the development of expert systems in the 1980s the rule-based approach was widely used to define domain and control knowledge. In the following sections the functionality and the properties of rule-based systems will be shown.

2.1. Principles of the Rule-Based Approach

Rule-based systems are a special form of knowledge-based systems. The rules in this knowledge-based systems are called *production rules*, *productions* or *condition-action rules* and have typically the following form [18,19]:

```
IF <conditions> THEN DO <actions>
```

Besides the definition of causal relations between objects the strongness of this approach is to describe and evaluate heuristic dependencies. Each rule is an independent knowledge unit and will be interpreted and executed by a domain independent rule engine. *Data-driven* rule engines match the conditions of rules with respect to existing data and identify rules to be executed next (*forward*

chaining). *Demand-driven* rule engines are focused on the action parts of the rules. Rules whose action part contains a previously defined goal will be collected in a conflict set. Out of this conflict set a rule is picked to be executed next. The conditions of this rule are the goals to be used in the next turn (*backward chaining*). In case of conflicts, which means multiple rules are able to be executed, the ordering of rules to be executed may be generated domain independent or with respect to domain dependent knowledge. A domain independent rule selection may consider the most current or the most special rule. A domain dependent selection may support priorities, goal functions or meta rules for example.

The principle of forward chaining rule-based systems is to look out for rules whose conditions are satisfied. In simple implementations in every turn all condition parts of all rules have to be checked against the current (new input) data. Because this is problematic with large rules bases optimizations are applied to make rule-based systems more efficient:

- Reduction of the data to check: In every turn only a (small) part of the data is changed, so in the next turn only these changes will be checked.
- Reduction of the conditions to check: The conditions of different rules are generally not disjunct, for identical parts checking has to be done only once.

These are the principles of the *Rete algorithm* [20], which was developed by Charles Forgy at the Carnegie Mellon University for the OPS-5 language (see above). Based on the conditions of the given rules a decision network in form of a data flow graph is generated ("rete" is Latin for *net*). Shared conditions will be checked only once, no matching is done twice. Changes of the data are propagated throughout the decision network with low costs. The algorithm is an efficient implementation of the rule matching mechanism. Many systems similar to XCON have been build using the same underlying technology. Variations of the Rete algorithm are still the core of most current rule-based systems.

2.2. Current Rule-Based Systems

In the following a quick overview over some examples of current rule-based systems, also known as *business rules management systems* (BRMS) is given:

2.2.1. IBM WebSphere ILOG JRules

ILOG JRules [21] is part of IBM's application and integration middleware "WebSphere" and has a market leading position in enterprise software for building and deploying rule-based applications for Java, mainframe, and SOA-based environments. The ILOG JRules framework is a software suite and comes with a set of tools, various rule languages, a rule repository including version control, a specific application server called *business rule execution server* and a rule engine, based on a variant of the Rete algorithm. The tool set includes tools for syntax and consistency checking of rules (broken and redundant rules). The rule languages of ILOG JRules differ from natural language syntax to Java programming language. Authoring, testing and deploying of business rules can be done by using the Eclipse IDE. Because of the Eclipse IDE integration developers can write and debug Java code and business rules within a single environment. Rules are

debugged in the same way as in Java by setting breakpoints in both application code and rules, stepping from one to another. In addition there exists a separate rule builder application as standalone and as web application, intended for the usage through business analysts rather than developers. Besides JRules for Java IBM also offers BRMS solutions supporting COBOL, .NET and z/OS (IBM mainframe) environments.

2.2.2. BOSCH Software Innovations Visual Rules

Another commercially successful BRMS is the Java-based Visual Rules [22] from BOSCH Software Innovations. Visual Rules is integrated in the Eclipse IDE and comes with a visual modeling approach. Basically a sort of program flow chart, called "rule tree", can be modeled. The modeling tool focuses on supporting the modeling of flow rules and decision tables in an as easy understandable and as intuitive way as possible (modeling rule trees with intuitive symbols, processing from left to right, from top to bottom, rules are modeled in sequence, etc). It allows rules and rule models to be hierarchically structured by using rule packages. In addition the modeling tool concentrates on reliable and extensive documentation of the rules and also on reporting mechanisms to present the results. An integrated test editor allows to define test cases for rules, the editing of test data and the expected results and the configuration and execution of tests. Furthermore Visual Rules covers all component areas of modern BRMS and consequently comes with a centralized rule repository supporting versioning, database integration for accessing data stored in a DBMS, and an execution server. The execution server allows the deployment of rules as web services (WSDL) to be used in SOA environments, e. g. for decision services in BPM and the integration of rules into .NET applications.

Interestingly Visual Rules is a non-Rete-based system, also called *explicit rule engine* or *explicit system*, which means the dependency of rules is explicitly specified in the modeler. The rules are then the basis for generating Java code automatically (COBOL is also supported). This code contains the modeled business logic and can simply be deployed from the modeling tool to the execution server or may be used in business applications direcly or via EJB (*Enterprise Java Beans*). In this way Visual Rules is a graphical modeling tool with a forward engineering approach. Because the rule trees modeled with Visual Rules contains a fix ordering of the rules this approach is well suited for applications where the rule tree is not changing much over the time and where it is relatively small-sized to be manageable. Applications with more dynamic and larger rule-bases are quite better handled by Rete-based rule engines.

2.2.3. Red Hat JBoss Enterprise BRMS

The JBoss rules platform is a Java-based open source BRMS including the "Drools" open source rule engine, which is commercially distributed and supported by Red Hat as part of the JBoss Enterprise BRMS [23]. It provides a logically centralized repository for storing and versioning the business knowledge, Eclipse IDE integration and a web-based environment allowing business users to view and (within certain constraints) update the business logic directly. The

enhanced debugging in Eclipse extends the normal Eclipse debugging with specialized views, e. g. for showing the currently running process instances and the state they are in. The deployment of business rules is possible with the JBoss application and SOA platforms as well as with third-party application servers and SOA environments (IBM WebSphere for example, see above). Additionally, the business rule engine may be deployed in standalone mode for direct application access without any server component. Moreover the JBoss Drools community version provides components for workflow and business processes, event processing, temporal reasoning, and automated (optimized) planning.

The Drools rules engine supports integration in various programming languages (e. g. Java, Python, Groovy) and supports the .NET platform and SOA environments. The representation of business rules ranges from the XML-based *Drools Rule Language* (DRL), "native" XML to decision tables (the last may be authored by external spreadsheet processing programs). The DRL is extendable and provides support for defining business rules in domain-specific, natural language. Core of the rule engine is *Rete-OO*, an enhanced implementation of the Rete algorithm tailored for object-oriented forward-chaining systems.

2.2.4. Jess

The well-known and often as *expert system shell* referred Jess [24] is a rule engine and scripting environment for the Java platform. It was written by Ernest J. Friedman-Hill at Sandia National Laboratories. Jess is not a BRMS at all. It can be used to build applications that use knowledge in form of declarative rules. Jess provides rule-based programming in Java by continuously applying a set of rules to a set of facts. The rules can modify the facts or execute any Java code. Jess was originally planed as a Java clone of the CLIPS rule language, a classical software tool for building expert systems like Charles Forgy's OPS language (see above). Like CLIPS the rule language of Jess is similar to Lisp syntax. Today Jess is a superset of the CLIPS programming language and still compatible, in that many Jess scripts are valid CLIPS scripts and vice-versa. The latest release of Jess includes a major update of the rule engine and supports its own declarative XML-based rule language called *JessML*. Furthermore it comes with a graphical rule development environment based on the Eclipse IDE providing editor functions, code formatting, error checking, run and debug commands and a graphical debugger for Jess programs. Jess is one of the rare "hybrid" systems allowing forward chaining as well as backward chaining. Like CLIPS, Jess uses an implementation of the rete algorithm. While CLIPS is open source, Jess isn't, but Jess is free for educational and government use.

On the basis of Jess the JSR-94 standard was founded: the Java Specification Request for a Java Rules Engine API [25]. ILOG JRules and JBoss Drools also support this API to allow the customer to be independent of a certain rule engine implementation.

2.3. Discussion on Rule-Based Systems

Current BRMS are typical rule-based systems with the well know advantages and drawbacks. The variations and improvements of the Rete algorithm are efficient,

but because of the weakness of the rule-based approach (maintenance problem, weakly structured) it is only suited for local dependencies with heuristic knowledge. The non-Rete-based system with explicit rules is a forward engineering modeling tool. It is convenient for rather small-sized and static rule trees.

Current rule-based systems come with enhanced capabilities for analysis and debugging improving the development and the maintainability of the rule base, an improved architecture storing rules separated from the domain knowledge, but the essential weakness of rules still remains due to characteristics of the rule-based approach (see above). The efficiency of the usage of todays rule engines in business process management depends on how rules are applied. The modeling of the process engineer, represented by the business process model, and the size of the rule base are essential for the efficiency of the approach. To manage the maintenance problem rules should only be applied in local contexts. Used as the exclusive inference mechanism, rules in current BRMS will become as critical as in the past, if the rule base grows up.

BRM is often applied in ERP environments, where a similar effect can be observed: During configuration of widely implemented systems like SAP ERP many parameters have to be customised. These parameters are input values for relations internally represented and evaluated like rules. As in rule-based systems these values have direct consequences for the control strategie of processes. The manpower and the high costs needed for the implementation and maintenance of these systems is definitely due to the comprehensive approach of ERP. It is also clearly an evidence for the complexity of the dependencies and that rules are applied too much, so it is hard for employees and consultants to maintain the overview of all details of the customized system.

It was one lesson of the 1980s to use different inference mechanism in knowledge-based systems. Different kinds of knowledge need different approaches to represent it. Each kind of knowledge should be modeled with an adequate representation mechanism. Besides this for existing dependencies appropriate solution mechanisms are needed. The requirement in software systems is to support this mechanisms. For the appropriate modeling of existing dependencies in BPM an additional inference mechanism is reasonable which does not compete against the process engine for the control strategy.

3. Constraint-Based Systems

In the area of artificial intelligence (AI) constraints have been in the focus of intensive research for decades [26,27]. There exist efficient algorithms and heuristics for the reduction of problem size and for an efficient generation of solutions.

3.1. Principles of the Constraint-Based Approach

Constraint techniques can be used to guarantee that specific relations hold, so the principle is a declarative paradigm. In general for the processing of constraints the problem is formulated as a *constraint satisfaction problem* (CSP). A CSP is a triple (V, D, C) where V denotes a finite set of variables, D denotes a set of

associated domains with possible values for each variable, and C denotes a finite set of constraints:

Definition 3.1 *A constraint satisfaction problem is specified by a triple* (V, D, C)*, whereas* $V = \{v_1, \ldots, v_n\}$ *is a finite set of variables with associated domains* $D = \{D_1, \ldots, D_n\}$ *with* $\{v_1 : D_1, \ldots, v_n : D_n\}$*. C is a finite set of constraints* $C_j(V_j)$*,* $j \in \{1, \ldots, m\}$*, whereas every constraint* $C_j(V_j)$ *sets a subset* $V_j = \{v_{j_1}, \ldots, v_{j_k}\} \subseteq V$ *of variables in relation to each other and restricts their valid combinations of values to a subset of* $D_{j_1} \times \cdots \times D_{j_k}$*.*

Each constraint defines a relation between a subset of variables and constrains the possible values for the involved variables. A constraint concerning only one variable is a unary constraint, constraints concerning two variables are binary constraints, constraints with three variables are ternary constraints, etc.

If for any reason the domain of a variable is reduced to a smaller number of values, this domain modification can be propagated through the constraint net using the available constraints to determine smaller value domains for the rest of the involved variables. Constraint propagation is used to reduce the problem size of CSPs and to reach different levels of local consistency with respect to the value domains of the constraint variables. Another aspect is the application during search algorithms looking for solutions of a specific CSP.

A short example is the following simple constraint problem: Let there be two variables a and b each with the assigned value domain $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and the binary constraints $c_1 : a + b = 10$ and $c_2 : a - b = 2$. The solution for this simple constraint problem would be $a = 6$ and $b = 4$. Note that besides arithmetic domains also symbolic domains are feasible and that the principles are not restricted to discrete domains.

Algorithms for constraint satisfaction are usually combined in a software component named *constraint solver*. Constraint solvers are used adopting the *black box* principle which makes integration easier and also allows the substitution of a solver with another solver.[1] Additionally the black box principle guarantees the separation of domain knowledge and control knowledge, because the control strategy is exclusively managed by the application (e. g. a process engine) using the solver component. For this the application gives relevant data (domain knowledge) as input to the constraint solver. The result of a solving process is output data that is updated values for the given input data. The application knows nothing about the internal solving process of the constraint solver and vice versa. So it is at the application to use the results of a constraint solver, and it is modeled in the application if and how the results will influence the further control strategy.

3.2. Constraints in Business Process Models

The enhancement of business processes with constraints should be realized in a straightforward manner to maximize the acceptance of the technology. To separate control knowledge from business rules the control flow is managed by a pro-

[1] Analogous to JSR-94 for Java rules engines, JSR-331 was founded: the Java Specification Request for a Constraint Programming API [28].

cess engine. Rules defining dependencies and relations in business processes are enhanced or replaced by constraints. As less as possible control knowledge should be modeled with rules to avoid the problems discussed in section 1. Currently we see the following fields of application for constraints in BPM:

- Constraints as a replacement for business rules, to bring control as soon as possible back to the process engine. The user is able to model the control flow intuitively and graphically as process model instead of abstract business rule definitions.
- Classical application of constraints, i. e. dependencies and relations of elements and attributes of a business process may be modeled as constraints to inference values due to specific (user) input and to guarantee a consistent process configuration. A constraint violation will result in a sort of exception handling and the execution of a (regular) process defined in the process model.
- Constraints as an instrument of QA: Known dependencies can be additionally made explicit through respective constraints in the process model to support further modeling ensuring the consistency of the process model. In further revisions of the process model the user will get feedback in case of an inconsistent modeling or modeling errors.

For the above cases we have generally two situations for the application of constraint technology in business processes:

- *Static* use of constraints at modeling: Constraints are used to check for a consistent process model.
- *Dynamic* use of constraints at runtime (simulation or live): Constraints are used to check for consistent states of process instances at runtime.

For the static use of constraints as modeling support in a process editor the user has to define known static dependencies in the process model as constraints. A constraint solver checks efficiently for inconsistencies during the modeling. If a constraint does not hold because of an inconsistent modeling the user will get a notification.

For dynamic use the constraints have to be applied in a simulation or in a real life system. Dependencies defined for dynamic use can only be checked at runtime because they require elements or parameters which will not be known before concrete process instances are avaiable. The constraint solver is checking for inconsistencies during the execution of the business processes. Inconsistencies will result in pre-defined actions.

Furthermore the process engine has to be enhanced by a component that monitors the constraints and reacts in case of constraint violations. We propose a *constraint handler* to perform these tasks. The constraint handler decides what do to if a specific constraint does not hold. The following alternative actions may be taken:

- A predefined process is executed (handling the constraint violation) parallel to the origin process, who triggered the predefined process and which execution is *continued*.

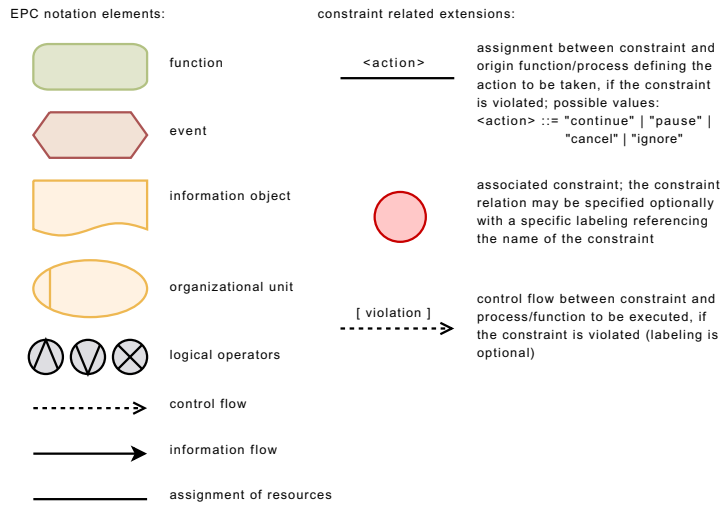

Figure 1. Legend of EPC notation elements and constraint related extensions used in this work.

- The origin process is *paused* until a predefined process is executed and finished.
- The origin process is *canceled* and (optionally) a predefined process is executed.
- The constraint violation is *ignored* and the execution of the origin process *continued*, for information only (optionally) a feedback may be generated and given to the user (modal or non-modal).

Besides this, the last action implies it, additional actions for user interaction like graphical user feedback, log messages, file output, email messages, or user-defined actions or applications may optionally be triggered by the constraint handler.

In order to realize the above we need the possibility to reference elements of a process model in a constraint. Constraints may constrain the structure (e. g. the ordering sequence) inside a process model, the type of model elements, attributes of model elements, and input/output data of processes. A graphical visualization is given in figure 1. In this work we use the notation of *event driven process chains* (EPC) to represent business processes [29]. In the figure only a subset of EPC notation elements relevant for the examples on the next pages are shown. In addition figure 1 contains the constraint related extensions used in the examples in this work.

To define relations on business processes a domain-specific language (DSL) [30] for constraint relations has to be specified. This constraint language has to allow the process designer to reference elements of process models and to specify the constraints themself. It is necessary to reference processes, attributes and input/output parameters of processes as well as defining operators on these elements (arithmetic operators like $=$, $\neq$, $<$, $>$, $\leq$, $\geq$, and temporal operators as introduced in [31] like `before`, `overlaps`, `during`). A smart way to realize this referencing is an explicit mapping of constraint variables to elements of the

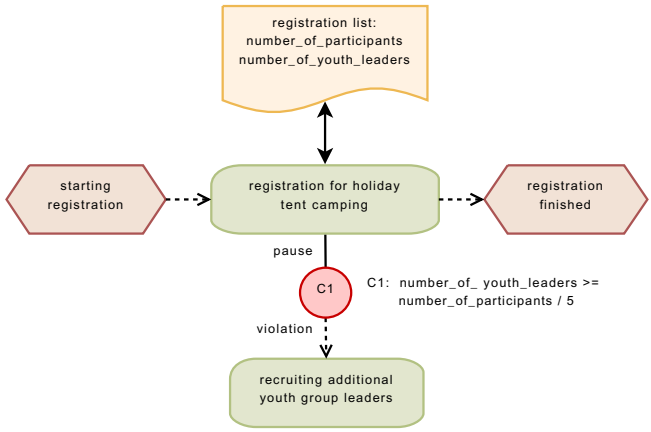

Figure 2. Constraining the number of youth group leaders in relation to the number of participants with a *value constraint*.

process model via *constraint connectors* [32]. The advantage of this approach is the possibility to use domain independent and replaceable constraint solvers.

In the following section we will show examples for constraints in business process models. Examples of different constraint classes, constraint relations and operators are given.

4. Examples

In this section constraint examples are embedded in business processes modeled as *event driven process chains* (EPC) [29]. To be generally understandable the graphical examples relate to the holiday tent camping scenario of a children's group from the first section of this work.

Even though natural language is the preferred representation in a customer ready final product, in the following the relations of constraints are assumed to be specified in a formal way (e. g. equations, inequations). Internally this is the necessary representation processable by the computer. Later on, e. g. in a user interface, a user-friendly and intuitive representation may be used on top of the internal representation.

4.1. Value Constraints

These constraints relate to the instance level of a process model. They constrain the values of the attributes of model elements which are provided as (input or output) parameters of a process instance.

In figure 2 an excerpt of the tent camping example is illustrated, determining the required number of youth group leaders which have to take care for the participants, i. e. the children of the holiday tent camping. During registration the number of participants and the number of youth group leaders are determined from the entries of the registration list. A constraint `C1` is associated with the registration process and constrains the number of youth group leaders in relation

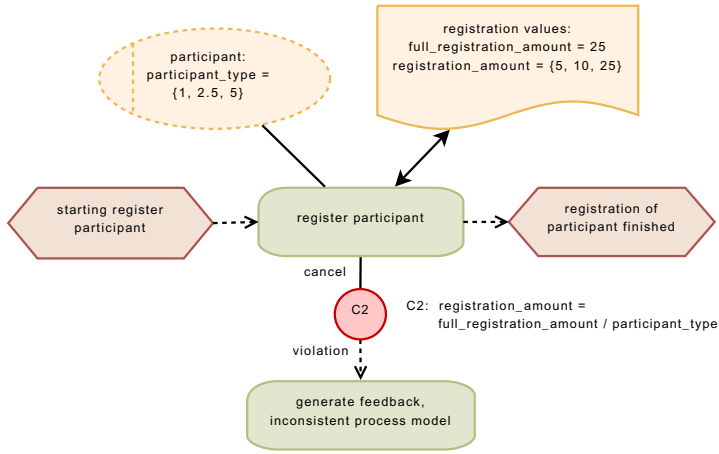

Figure 3. *Value constraint* determining the registration amount for different classes of participants.

to the number of participants. If the constraint is violated (this means the number of registered youth group leaders is too low) the registration process is paused and another process for recruiting additional youth group leaders is triggered. The origin registration process is continued when this recruiting process is finished. Notice that we assume, that the constraint has access rights for all the data the process also has access rights for.

Another example of a value constraint is illustrated in figure 3. It contains the process of the registration of a single participant. The constraint `C2` is used to determine the particular registration amount for different classes of participants (incl. youth group leaders): A participant has a specific type representented by a numeric value in the attribute `participant_type`. For example a child is represented by the value `1`, a youth by the value `2.5` and a youth group leader by the value `5`. If a concrete participant is to be registered, the concrete value is fixed either to `1`, `2.5` or `5`. In other words the domain of the constraint variable `participant_type` is reduced to a single value. This reduction leads to another domain reduction: If the constraint `C2` is propagated the value for the concrete registration amount is determined. So the value of the variable `registration_amount` for a child is `25`, for a youth it is `10` and for a youth group leader it is `5` after the propgation of the constraint.[2] As a result we can say constraints allow compact modelings of decision processes to determine specific values in process models at runtime.

Notice that besides this calculation of a concrete registration amount the constraint in figure 3 is used for QA reasons: If for any reason the constraint is violated, that means there are no compatible values in the domains of the constraint variables to fulfill the constraint, this indicates an inconsistent process

[2]The amount for youth group leaders is lower than for the children in this scenario, because their work is to supervise the other participants. They are assisted by the youth, who also look after the children and whose amount is lower, too.

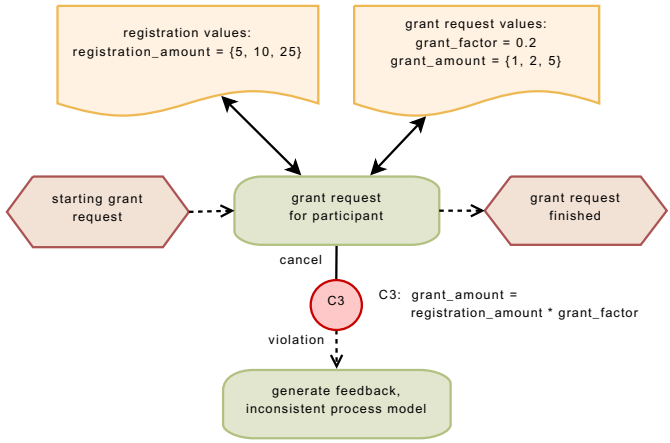

Figure 4. Using constraint propagation of *value constraints* to determine the grant amount per participant.

model and by this a modeling failure. The constraint may be used to generate user feedback in this case. The origin process has to be canceled.

In figure 4 the process of a grant request for a single participant is shown. In combination with the constraint example in figure 3 the propagation of a domain reduction over different constraints and processes can be observed: The constraint `C2` in figure 3 reduces the domain of `registration_amount` to a single value when a concrete participant is registered. This domain reduction is propagated by the constraint `C3` in figure 4 by using a given `grant_factor` to determine the granting for different participants. If the participant is a child (registration amount: `25`), then the grant amount is propagated to `5`. For a youth (registration amount: `10`) the grant amount is propagated to `2`, and for a youth group leader (registration amount: `5`) the grant amount is propagated to `1` in this example.

Notice that the constraint `C3` cancels the execution of the origin process in case of a constraint violation. So as `C2` the constraint `C3` is used for QA reasons, too, and it may generate a user feedback if the constraint is not fulfilled.

Another variant of a value constraint is illustrated in figure 5. In the scenario of a summer holiday tent camping the bottled water in stock is an important resource avoiding thirsty children. In the example a resource constraint `C4` is used to check the number of available boxes of bottled water. Possible values are in the (closed) interval between 0 and 20. If the number of available boxes of water bottles is equal to or below a specific value (here: 3), then an ordering to the local beverage supplier has to be initiated. The origin process has to be continued.

The resource related constraint `C4` in figure 5 is a "global" constraint, valid during almost the whole scenario. It has to be continuously checked during the process `perform holiday tent camping` and all of its sub-processes.

4.2. Model Constraints

Another sort of constraints in business processes is the category of *model constraints*. They are related to model elements and therefore the structure of a

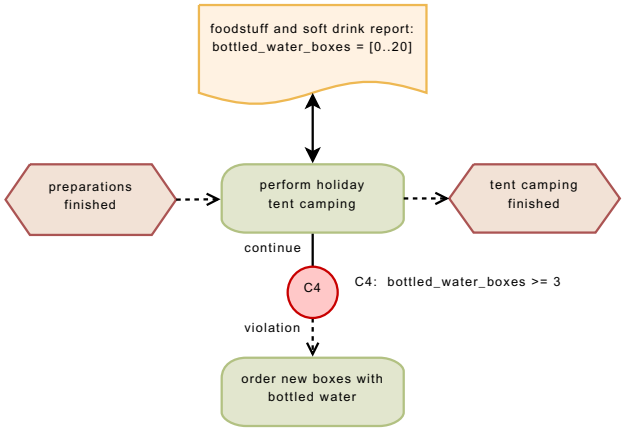

Figure 5. Example for a resource related *value constraint*, triggering an ordering action.

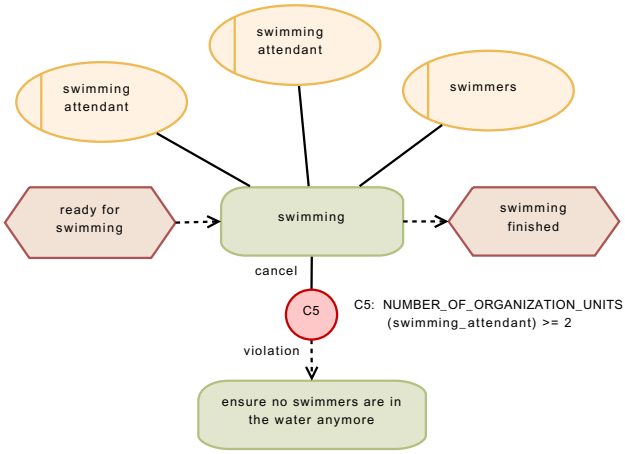

Figure 6. *Model constraint* ensuring a minimum number of swimming attendants taking care for the swimming children.

process model. Therefore model constraints relate to the conceptual level in process modeling. According to this it is necessary to reference characteristics and properties of model elements by model constraints. These properties include the following elements related to a specific process:

- number and type of (input/output) parameters
- number of processes (e. g. sub-processes or a section between specific elements) and type of processes
- number and type of used documents/data resources
- number and type of involved organizational units

Considering the holiday tent camping scenario in figure 6 an example of a model constraint is given. An operator `NUMBER_OF_ORGANIZATION_UNITS` is used to get the number of associated swimming attendants, taking care for the swimming children. There is a minimum number of 2 swimming attendants necessary, so the

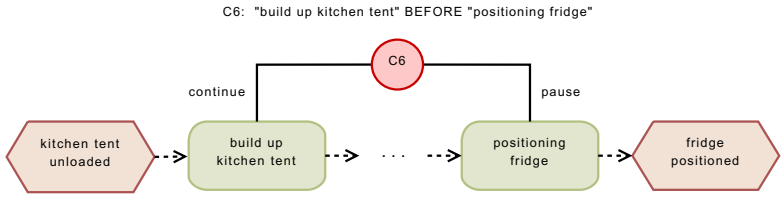

Figure 7. *Temporal constraint* determining the sequential ordering of processes.

constraint `C5` is not violated. If the number of associated swimming attendants in the process model is below this value, then the `swimming` process is canceled and a new process is started to ensure no swimmers are in the water anymore.

Model constraints are usually applied for QA reasons. The modeled knowledge is redundant with the existing information in the process model. The constraint is specified by the modeler to ensure a specific relation holds in the current revision and in future revisions of the process model. In this way future revisions of the process model will be consistent with the knowledge explicitly modeled by the constraint.

In figure 6 the number of swimming attendants is ensured by the constraint `C5`. In figure 2 the constraint `C1` determines the number of youth group leaders. While `C1` is a value constraint (instance level) the constraint `C5` is a model constraint (conceptual level). The difference is that `C1` refers at runtime to a value of a data resource (`registration list`). The constraint `C5`, in contrast, refers to the modeled organization units in the process model.

4.3. Temporal Constraints

Temporal constraints are a specific form of value constraints related to temporal aspects at the instance level of a process model. With temporal constraints techniques known from temporal reasoning using constraint propagation can be applied to business processes. The *temporal constraint satisfaction problem* (TCSP) is used for planning and scheduling and uses its own temporal logic to represent temporal relations [31]. In our approach temporal constraints are used for flow control instead for scheduling.

In figure 7 a temporal constraint supporting the process engine managing the control flow is shown. The temporal operator `before` ensures that a specific process has to be finished before another process may start. In figure 7 regarding the tent camping scenario the constraint `C6` ensures, that the positioning of the fridge in the kitchen tent is not able to start until the kitchen tent is completely built up. Otherwise, if the process engine is going to start the positioning process (this means the constraint is violated), the positioning process is paused, the process to build up the kitchen tent is continued. A reporting or feedback is not necessary in this example.

In a simple sequential ordering as in the example in figure 7 the constraint may be be added for QA reasons. Making those semantic dependencies explicit with constraints will avoid incorrect process models in the future. In particular this affects process models which are often modified and/or which are modified by different process modelers.

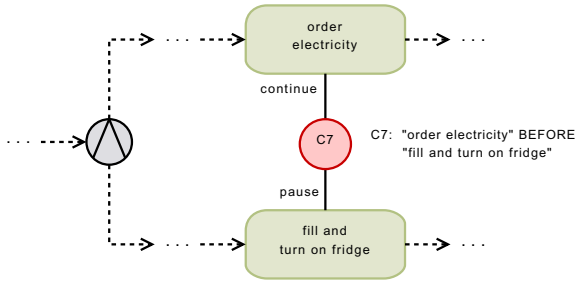

Figure 8. Constraining the control flow in a concurrent situation with a *temporal constraint*.

Another example of a temporal constraint is illustrated in figure 8. The control flow in this example is split at the `AND` operator. Therefore we have a concurrent situation. Using a temporal constraint allows to determine the execution time of processes in parallel branches of the process model. The `before` operator may also be used here to prevent a process to start before another process has ended. In the example the constraint `C7` ensures the fridge will not be filled and turned on until electricity is ordered. The ordering process is continued in either case. The process of filling and turning on the fridge is paused if the ordering process is not finished yet.

5. Discussion

Decision support in BPM can be achieved by the widely used business rules or by constraint-based approaches. Assuming control knowledge explicitly modeled with graphical process models is more intuitive to the user and has a significant better maintainability than declarative approaches, we propose to enhance process models with constraints instead of rules. The enhancement of business processes with constraints as declarative mechanism avoids the lack of separation between domain knowledge and control knowledge. Constraints allow to use the process engine exclusively for defining the control strategy.

5.1. Constraints and Processes: "Graphical" Rules?

The presented approach and the given examples may let guess someone that constraints in combination with business processes is like using rules "graphically": The constraint seems to be the condition part and the process to be executed in case of a constraint violation seems to be the action part of a rule. The difference is that rules would capture the control flow from the process engine. In our approach in contrast the process engine resumes the control flow if a constraint is violated. If constraints in combination with processes are used like business rules with a significant effect on the control flow, they should be used as careful as business rules: limited application with sense of proportion and only in local contexts. Furthermore the usage of constraints as a replacement or "simulation" of business rules is only one aspect of the application of constraints in combination with business processes. In addition the classical inferences and the field of QA are grateful fields for the application of constraints in the area of BPM.

5.2. Constraints versus Rules

The usage of rules differs generally from the usage of constraints. While rules contain a condition part and an action part (which is executed when the condition part is fulfilled) a constraint is assumed to always hold. In that it can be observed that a rule has a direction while a constraint is non-directional. Also with respect to the model elements and attributes related by a rule there is clearly a direction defined: The attributes of elements related in the condition part of a rule have a direct effect on the model elements related in the action part, but *not* automatically vice versa. This is generally different for constraints: Here the constraint solving algorithms check if the relation defined by the constraint still holds or not. A binary constraint e. g. is checked for both constraint variables, so there we have a birectional evaluation of the constraint and also bidirectional effects on the model elements. For n-ary constraints there will be a n- or multi-directional evaluation.

Another difference between rules and constraints is the fact that rules are usually executed only one time while constraints have to be checked permanently for violations. Furthermore constraints are really a declarative formalism because they are totally free of control structures. The ordering for checking constraints is not relevant for the result (black box). This is an important benefit which got the respective appreciation practice, e. g. in the domain of knowledge-based configuration: "The knowledge engineer is relieved from thinking about evaluation of a constraint. Even the dependencies between several restrictions are considered by the constraint net. This is a great advantage over rule-based systems." [33].

5.3. Applying Constraints in Which Contexts?

In general the application of constraints is reasonable if (1) multiple elements are in relation to each other, (2) there exist alternative values in the value domains of these elements, and (3) these alternative values may be reduced by propagation so we can get a domain reduction and thus a problem reduction as result. Examples related to business processes where this aspects come together may be found in section 4.

Constraints should rather not be applied if an aspect has to be checked only one time and this checking results in an concrete control decision. This work may be better done by a business rule. Constraints are an adequate technique if checks have to be done permanently, e. g. during the execution of a complex business process (and all of its sub-processes).

Also constraints should not be used if the same aspect can be modeled as a simple decision using the respective control flow operators in a process model. Constraints however allow compact modelings of decision processes to determine specific values in process models. They are well-suited to model complex decisions and relations over multiple processes. Rules in contrast should be applied limited and only in local contexts. They are suited to represent heuristic knowledge, which is executed only one time.

5.4. Complexity

The above differences between rules and constraints result in a higher complexity of the constraint technology. Because of this constraint-based systems are often criticized to be inefficient managing large constraint nets. Despite there exist a couple of efficient (i. a. heuristic) algorithms handling different kinds of constraint problems the fact is less relevant in this case at all. In the BPM approach described in this paper the focus is on the control flow in process models managed by a process engine. Constraints are used additional to BPM techniques. So there is less the problem of complexity of constraints as in the useful enhancement of BPM with constraints. In contrast to constraint-based systems, where constraints are the sole knowledge representation technique, supporting BPM we have only simple constraints. Also the complexity of the constraint net is about a manageable size, because the control flow is mostly determined by the process model. The constraint technology is used to provide a flexible way to react on violations of know limitations and restrictions and thus supporting the process engine executing the process model.

5.5. Declarativity

A point of critique often heard concerns the declarative paradigm, constraint technology is based on. Constraints are a declarative form of knowledge representation, which is often not intuitive and transparent to users thinking in procedural categories. Here also holds the above argument, that constraint technology is just enhancing process models in BPM. Different from pure constraint-based systems the control flow is defined procedural by process models. Constraints (with limited scope for example) provide an intuitive way to define restrictions and limitations directly (e. g. graphically) inside a procedural process model, not stated "standalone" as an abstract and declarative constraint problem. Processes triggered in case of constraint violations are directly executed by the process engine. So constraints are a flexible and declarative formalism, allowing powerful shortenings in process models without taking control from the process engine. But like any other knowledge representation constraints should be used with sense of proportion, excessive usage is not suggested.

To get the point, business rules and enlarged rule bases will lead to maintainability problems. Rules are flexible, but intransparent. Because of the declarativity of rules it is not stated explicit at which time a rule is executed and so the control flow is not intuitive clear. In BPM the control flow is stated explicit in process models. Enhancing these process models with constraints provides a dynamic and declarative component without defining control knowledge outside the process model.

6. Related Work

This paper proposes the application of constraint satisfaction to ensure the consistency of relations in business processes. Related work to this approach therefore

may be found in the domain of constraint satisfaction [26,27]. Existing constraint satisfaction approaches may be employed to achieve the goals outlined in this paper.

In [34] the definition of the *open constraint satisfaction problem* ist given, a branch of constraints research related to distributed CSP and collaborative agent environments. An additional set of constraints is used to capture external information. An example scenario is given, illustrating the application in business process modeling in distributed environments, where a constraint solver does not necessarily know any of the external constraints held by other agents. The author then sets the research agenda identifying some important research areas in open constrained optimization research. In [35] the open constraint satisfaction model is used for distributed scheduling. On distributed CSP many studies are available in research literature [36].

The authors in [37] show their idea how to extend EPC diagrams with constraints in distributed environments. They introduce a new model element called *process module* to encapsulate single functions or sub-processes separated from the surrounding process. Process modules are used to hide process information of different corporations in collaborative business environments. They may be enhanced by pre- and postconditions, which are to be modeled like additional events. A constraint is associated with a nondirectional edge to a process module. Constraints are used to document the limitations during the execution of functions and sub-processes covered by a process module.

Temporal constraints are introduced by [31]. The syntax and semantics of temporal operators is specified. These operators are used to define (constraint-)relations over time intervals. A constraint propagation algorithm is presented to propagate temporal constraints. A survey over different representations of time and basic techniques is given in [38]. The application of temporal constraints in BPM is presented in [39], where the modeling and execution framework for business processes with the focus on scheduling constraints is specified. Business processes are modeled as a *business process constraint network* using temporal constraints to define the relations between different tasks. All tasks are represented by time intervals. Tools with integrated temporal constraint networks already exist, e. g. as workflow management systems [40].

An approach combining business rules and constraint techniques is presented in [41]. Business rules has to be formulated as statements in *Rules2CP*, a general purpose rule-based modeling language [42]. These business rules are translated into constraint programs, which can be efficiently tackled by constraint solvers. The aim is to make constraint programming technology easier to use by non-programmers and without deep understanding of the underlying technology. This is also in the focus of *OpenRules* [43], a *Business Decision Management System* combining the usage of rules and constraints: In [44] a constraint-based approach for the implementation of rule engines is presented that is able to execute *rule-based decision models*. Business users may use their favorite interfaces like Excel-based decision tables or BRMS rule editors. The developed *Rule Solver* generates a constraint satisfaction problem from this decision model and solves it with the same results as using a rule engine. Additionally it will generate solutions when business rules only partially define a problem.

There exist other approaches combining constraints and rules: In [45] constraints are used for the verification of rule programs. In contrast in [46] rules are used to define a control strategy for the evaluation of constraints.

7. Conclusion & Outlook

Business processes often use BRMS for decision support. Even business processes defined and controlled exclusively by business rules are common. In large system approaches using a sole declarative knowledge representation technique introduces serious problems. In BPM we suggest to model the control strategy explicit and as exclusively as possible as process models. Modeling with graphical process models is more intuitive and will result in a significant better maintainability than modeling declarative and compact representations, e. g. in form of decision tables. Process models may be supported by business rules for local decisions, but rules should not be used for global control. Using rule-based approaches introduces some serious problems, because rules contain domain knowledge as well as control strategy. Rules are declarative, weakly structured and difficult to manage and maintain. For large rule bases the user can not be sure if the problem is completely covered by the rules, modifications often result in unwanted consequences.

Instead of rules we propose a constraint-based approach. Constraints also have a declarative paradigma but avoid the lack of separation between domain knowledge and control strategy. Constraint solvers used as black box by a process engine will merely compute an output based on the given domain knowledge and return these as new input for the process engine. A constraint solver will give control as soon as possible back to the process engine. The process engine exclusively has to decide how to proceed. So constraints support a process engine without competing for the control strategy.

The challenges of this approach lies in the useful enhancement of BPM with constraint technology. This requires the specification of an appropriate API containing a constraint language to reference business process model elements defining relations on them. Furthermore constraints are an alternative for the business rules approach. The weakness of the widely used business rules can be moderated by constraint technology and respectively enhanced by constraints. So enhancing BPM with constraints may lead to process models containing both, constraint technology and also rules each in adequate contexts. Another field of application for constraint technology in BPM is the area of QA. Known dependencies in process models can be made explicit by additional constraints supporting further revisions by ensuring consistent modelings. Further research has to be done with respect to process hierarchies, the scope of constraints, the integration of solutions of sub problems, and meta constraint solving.

The next steps will be the preparation of case studies and examples for the application of constraint technology in BPM practice. This should preferably be supported by practitioners and experts in selected fields. The intention is to generate more input for requirements relevant in practice.

Acknowledgment

The author would like to thank Werner Kober and Petra Burmester for useful discussions and comments and Elke Pulvermüller for advice and detailed feedback.

References

[1] J. L. G. Dietz, "On the Nature of Business Rules," in *Proc. CIAO!/EOMAS 2008*, ser. LNBIP, J. L. G. Dietz, A. Albani, J. Barjis, W. Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, and C. Szyperski, Eds., no. 10. Springer, 2008, pp. 1–15.

[2] W. M. P. v. d. Aalst, A. H. M. t. Hofstede, and M. Weske, "Business Process Management: A Survey," in *Proc. BPM 2003*, ser. LNCS, W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, Eds., no. 2678. Springer, 2003, pp. 1–12.

[3] P. Harmon, "Business Process Management: Today and Tomorrow," in *Proc. BPM 2008*, ser. LNCS, M. Dumas, M. Reichert, and M.-C. Shan, Eds., no. 5240. Springer, 2008, pp. 1–1.

[4] P. Gilbert, "The Next Decade of BPM," in *Proc. BPM 2010*, ser. LNCS, R. Hull, J. Mendling, and S. Tai, Eds., no. 6336. Springer, 2010, pp. 1–4.

[5] W. Bandara, P. Harmon, and M. Rosemann, "Professionalizing Business Process Management: Towards a Body of Knowledge for BPM," in *BPM Workshops, BPM 2010 – Revised Selected Papers*, ser. LNBIP, M. Muehlen, J. Su, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, and C. Szyperski, Eds., no. 66. Springer, 2011, pp. 759–774.

[6] J. McDermott, "R1: A Rule-Based Configurer of Computer Systems," *Artifical Intelligence*, vol. 19, no. 1, pp. 39–88, Sep. 1982.

[7] W. P. Birmingham, A. Brennan, A. P. Gupta, and D. P. Sieworek, "Micon: A Single Board Computer Synthesis Tool," *IEEE Circuits and Devices Magazine*, vol. 4, no. 1, pp. 37–46, Jan. 1988.

[8] M. Golden, R. Siemens, and J. C. Ferguson, "What's Wrong with Rules?" in *Proc. WESTEX-86*, G. S. Robinson and M. S. Cook, Eds. IEEE Computer Society Press, 1986, pp. 162–165.

[9] E. Soloway, J. Bachant, and K. Jensen, "Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rule-Base," in *Proc. AAAI-87*. AAAI Press, 1987, pp. 824–829.

[10] V. Barker and D. O'Connor, "Expert System for Configuration at Digital: XCON and Beyond," *Communications of the ACM*, vol. 32, no. 3, pp. 298–318, Mar. 1989.

[11] D. Sabin and E. C. Freuder, "Configuration as Composite Constraint Satisfaction," in *Configuration – Papers from the AAAI Fall Symposium*, B. V. Faltings and E. C. Freuder, Eds. AAAI Press, 1996, pp. 28–36.

[12] D. Sabin and R. Weigel, "Product Configuration Frameworks – A Survey," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 42–49, Jul./Aug. 1998.

[13] J. McDermott, "R1: The Formative Years," *The AI Magazine*, vol. 2, no. 2, pp. 21–29, 1981.

[14] P. Harmon, "How DEC is Living with XCON," *Expert Systems Strategies*, vol. 5, no. 12, pp. 1–5, 1989.

[15] B. Neumann, "Configuration Expert Systems: a Case Study and Tutorial," in *Artificial Intelligence in Manufacturing, Assembly and Robotics*, H. O. Bunke, Ed. Oldenbourg Verlag, 1988, pp. 27–67.

[16] A. Günter and R. Cunis, "Flexible Control in Expert Systems for Construction Tasks," *Applied Intelligence*, vol. 2, no. 4, pp. 369–385, 1992.

[17] J. Bachant, "RIME: Preliminary Work toward a Knowledge-Acquisition Tool," in *Automating Knowledge Acquisition for Expert Systems*, S. Marcus, Ed. Kluwer Academic Publishers, 1988, ch. 7, pp. 201–224.

[18] P. Jackson, *Introduction to Expert Systems*, 3rd ed. Addison-Wesley, 1998.

[19] S. J. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach (The Intelligent Agent Book)*, 2nd ed. Prentice Hall, Dec. 2002.

[20] C. L. Forgy, "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, vol. 19, no. 1, pp. 17–37, Sep. 1982.
[21] IBM WebSphere ILOG JRules. [Online]. Available: http://www-01.ibm.com/software/websphere/ilog/
[22] BOSCH Software Innovations Visual Rules. [Online]. Available: http://www.visual-rules.com/
[23] Red Hat JBoss Enterprise BRMS. [Online]. Available: http://www.jboss.com/products/platforms/brms/
[24] Jess. [Online]. Available: http://www.jessrules.com/
[25] JSR-94: Java Specification Request for a Java Rules Engine API. [Online]. Available: http://www.jcp.org/en/jsr/detail?id=094
[26] E. P. K. Tsang, *Foundations of Constraint Satisfaction*. Academic Press, 1993.
[27] R. Dechter, *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
[28] JSR-331: Java Specification Request for a Constraint Programming API. [Online]. Available: http://www.jcp.org/en/jsr/detail?id=331
[29] A.-W. Scheer, *ARIS – Business Process Modeling*, 3rd ed. Springer, 2000.
[30] M. Fowler, *Domain-Specific Languages*, 1st ed. Addison-Wesley, 2010.
[31] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM (CACM)*, vol. 26, no. 11, pp. 832–843, Nov. 1983.
[32] W. Runte and M. El Kharbili, "Constraint Checking for Business Process Management," in *Proc. INFORMATIK 2009*, ser. LNI, S. Fischer, E. Maehle, and R. Reischuk, Eds., no. 154. GI, Sep. 2009, pp. 4093–4103.
[33] M. Kopisch and A. Günter, "Configuration of a Passenger Aircraft Cabin Based on Conceptual Hierarchy, Constraints and Flexible Control," in *Proc. IEA/AIE-92*, ser. LNCS, F. Belli and F. Radermacher, Eds., no. 604. Springer, 1992, pp. 421–430.
[34] E. P. K. Tsang, "Constraint Satisfaction in Business Process Modelling," *The Journal Of Management and Economics*, vol. 7, no. 7, Nov. 2003.
[35] E. P. K. Tsang, T. Gosling, B. Virginas, C. Voudouris, G. Owusu, and W. Liu, "Retractable Contract Network for Empowerment in Workforce Scheduling," *Multiagent and Grid Systems*, vol. 4, no. 1, pp. 25–44, Jan. 2008.
[36] M. Yokoo and K. Hirayama, "Algorithms for Distributed Constraint Satisfaction: A Review," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 2, pp. 185–207, Jun. 2000.
[37] R. Klein, F. Kupsch, and A.-W. Scheer, "Modellierung inter-organisationaler Prozesse mit Ereignisgesteuerten Prozessketten," in *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, A.-W. Scheer, Ed. Institut für Wirtschaftsinformatik, Nov. 2004, no. 178.
[38] J. F. Allen, "Time and Time Again: The Many Ways to Represent Time," *International Journal of Intelligent Systems*, vol. 6, no. 4, pp. 341–355, Jul. 1991.
[39] R. Lu, S. Sadiq, V. Padmanabhan, and G. Governatori, "Using a Temporal Constraint Network for Business Process Execution," in *Proc. ADC 2006*, ser. CRPIT, G. Dobbie and J. Bailey, Eds., no. 49. Australian Computer Society, 2006, pp. 157–166.
[40] M. Reichert, S. Rinderle, and P. Dadam, "ADEPT Workflow Management System," in *Proc. BPM 2003*, ser. LNCS, W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, Eds., no. 2678. Springer, 2003, pp. 370–379.
[41] F. Fages and J. Martin, "From Rules to Constraint Programs with the Rules2CP Modelling Language," in *Proc. CSCLP 2008*, ser. LNCS, A. Oddi, F. Fages, and F. Rossi, Eds. Springer, 2009, no. 5655, pp. 66–83.
[42] Rules2CP Modeling Language. [Online]. Available: http://contraintes.inria.fr/rules2cp/
[43] OpenRules. [Online]. Available: http://openrules.com/
[44] J. Feldman, "Representing and Solving Rule-Based Decision Models with Constraint Solvers," in *Proc. RuleML 2011*, ser. LNCS, F. Olken, M. Palmirani, and D. Sottara, Eds. Springer, 2011, no. 7018, pp. 208–221.
[45] B. Berstel and M. Leconte, "Using Constraints to Verify Properties of Rule Programs," in *Proc. ICST 2010*. IEEE Computer Society, Apr. 2010, pp. 349–354.
[46] Y. Caseau, "Constraint Satisfaction with an Object-Oriented Knowledge Representation Language," *Applied Intelligence*, vol. 4, no. 2, pp. 157–184, 1994.