Using Soft Constraints to Guide Users in Flexible Business Process Management Systems (BPMS)

Christian Stefansen*
Department of Computer Science (DIKU),
University of Copenhagen, Copenhagen, Denmark
E-mail: cstef@diku.dk
*Corresponding author

Signe Ellegård Borch
IT University of Copenhagen,
Copenhagen, Denmark
E-mail: elleborch@itu.dk

Abstract: Current BPMS allow designers to specify processes in highly expressive languages supporting numerous control flow constructs, exceptions, compensation handling, complex predicates, policies, etc. The resulting specification can be anything from extremely flexible to extremely rigid, but there is an unfortunate trade-off: rigid specifications become a hindrance when unanticipated deviations occur, and too flexible specifications lack information about best practices and recommended order, because everything is possible. Both specifications are expressed exclusively in terms of hard constraints and even very clever specifications can only partially alleviate this. When the process designer writes the process as described by the business analyst, inevitably important intentional information is either abandoned or promoted to hard constraints. We argue that hard constraints are insufficient, and that they lead to either rigid (dictatorial) or support-less (anarchistic) process specifications. Soft constraints can make this trade-off less painful. If process designers can write soft constraints directly in the business process specification, the BPMS can accommodate unanticipated deviations while having enough information about best practices to guide the user through the process. As a consequence we suggest strongly to prefer soft constraints over hard constraints. Soft constraints can specify not only that some rules can be violated, but also by how much and how serious a concrete violation is. The BPMS should allow designers to easily specify soft goals and allow its users to immediately see the seriousness of their violations at runtime. We believe this provides a desirable combination of guidance and flexibility that is not otherwise attainable. To achieve this soft goals must be made explicit in the process model and be understood by the BPMS. We outline how this is done using a multi-criteria optimization problem formulation, and suggest some changes to the design methodology employed by process designers.

Keywords: business processes; workflow; constraint specification; soft goals; business process flexibility; Pareto optimality.

Reference to this paper should be made as follows: Stefansen, C. and Borch, S. E. (2007) 'Balancing Flexibility and Support in Business Process Management Systems (BPMS)', *Int. J. Business Process Integration and Management*, Vol. *, Nos. */*/*, pp.**–**.

Biographical notes: Christian Stefansen is currently a Ph.D. fellow at the University of Copenhagen at the Department of Computer Science (DIKU). His research focuses on applying formal methods from programming languages (in particular, process algebra) and game theory to workflow, ERP systems, financial derivatives, and tax legislation. Signe Ellegård Borch is currently a Ph.D. fellow at the IT University of Copenhagen. She is doing empirical research on definitions, design criteria, and use of business process models in different contexts, such as workflow management systems, CSCW, and the development of ERP systems.

1 INTRODUCTION

The past five–ten years have witnessed significant changes in the way business processes management systems (BPMS) are perceived, as well as the context in which they take part. Whereas BPMS were previously thought of as support systems, the interest in business processes is now focused on systems that *actively execute* processes in collaboration with other processes, services, and humans.

The development that lead to BPMS taking the center stage was brought on by management evangelists, who solicited the successful business process perspective (Hammer & Champy, Davenport, Michael Porter). Later important legal changes took place. Most notoriously, the *Sarbanes-Oxley Act* was signed into law and now requires managers to oversee that companies directly document the processes that produce any data used in external reporting under penalty of law.

At the same time, as the Internet grew in significance, so did the pressure to make systems Internet-accessible and interoperable. The *service-oriented architecture* (SOA) became the enabling paradigm, and it was followed by numerous standards designed to make a shift to the new paradigm possible. Previously, integrating business processes across several systems and business entities had been cumbersome or just plain uneconomical, but as businesses gradually started the move toward SOA, this became much easier to achieve. This development can also be seen as an instance of a general move towards higher levels of abstraction. Successive generations of programming models tend to enable increasingly abstract entities to be represented directly; we have now reached the level where distributed business processes can be represented explicitly.

In short, the business climate, the regulatory climate, and the prevailing IT architecture have all become increasingly favorable to the process paradigm. This development has allowed the BPMS to take on a new role: in a setting where the individual steps of a business process are exposed as Internet services, the BPMS can be an *active orchestrator* that *executes* the business processes by delegation, rather than just serving as a *passive support system* deep inside an isolated part of the organization.

The change from passive support system to active orchestrator places new demands on the BPMS. In passive support systems the business process could be written quite stringently as a program, and if the real-world process happened to deviate from the system-prescribed one, this was not necessarily a problem. As long as the users knew what they were doing, they could ignore the system and report dummy task completions back to the system until the real-world process and the system-prescribed process were in agreement again. Today, because the system orchestrates the process, users will find it much more difficult to ignore or trick the system if the real-world process deviates from the prescribed one. Because the BPMS is what makes the process progress, the system process and the real-world process have to agree to a much larger extent than before.

If the processes are described and executed stringently, users will find them exasperating to work with when the real-world process sees variations that were not anticipated in the process design. A very typical pitfall is to only express the best practice in the process design meaning that the users have to come up with workarounds when best practice cannot be followed. On the other hand if the process designer tries to forestall such rigidity by making the process very flexible, there is usually no way of indicating to the users what the preferred practice is. The result is then a system that provides no *support* to its users. Crudely put, a good system should continuously give users an overview of what they *may* do, what they *should* do, and what they *must* do – and let the process designers easily describe these modalities in the process template.

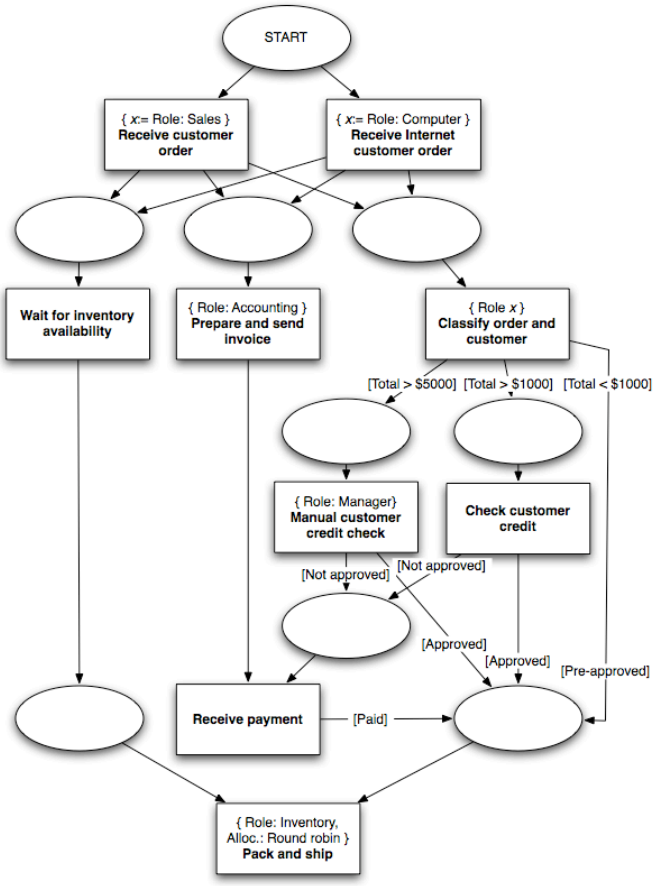

Figure 1 *Simplified order handling process*

An Example Process

Let us illustrate these points with an example. Figure 1 shows a simplified order handling process in simplified *Colored Petri Net* notation. Several elements are omitted such as data sources, data formats, timeouts, service connectors, and access control. The process takes a received order from the sales department or from the company's website and facilitates shipping, invoicing, payment, and, if necessary, credit approval. The process specifies a base policy that orders less than $1000 are pre-approved, orders of more than $1000 are subject to credit checks, and orders

over $5000 require a manual credit check by a manager. If the customer is not credit approved, advance payment is required.

Having $1000 and $5000 as delimiters for a base policy seems very reasonable, but yet it raises the concern: will we *always* adhere strictly to this policy? Presumably not. A customer order of $1001 should makes us think about bending the rule slightly, and conversely, we may wish to do an additional manual credit check on a $2000 order, if the customer is known or suspected to be a bad payer.

Hence this policy is clearly bendable. It is also clear that deviating from the policy is less problematic in the case of a $1001 order than in the case of, say, a $9500 order. Therefore the user should be allowed to violate this policy and make an informed decision, and in doing so, the system should somehow indicate how serious a violation is being made. Introducing an extra path in the process – call it *User override* – does not adequately address the issue, because it does not measure the gravity of the violation; it merely records the fact that a violation has taken place. Also the existence of such an extra path does explain to the user what is the preferred path or what would be needed to justify an override.

To sum up, removing the credit policy will deprive the process specification of useful guideline information, but making it a hard constraint will render the users unable to make exceptions even in very reasonable cases. Something in between is needed: we need a measure of the *soft goal*: what is our *credit risk*? Violating the rule once will not seriously damage the company's goal to minimize credit risk, but if we constantly need to violate the credit policy, company performance on the credit risk goal will reflect this, and management will be able to take appropriate steps to rectify the situation.

Now consider the sequential constraint that *Pack and ship* can occur only when *Wait for inventory availability* is complete. Should this constraint be flexible? Definitely not. The constraint is a hard constraint, because shipping a good that one does not have in one's possession amounts to violating the laws of physics. Here the BPMS must stand firm on its constraint. (For the sake of argument, ignore the possibility of partial shipments. Our discussion will argue that generally there are many unanticipated exceptions even to hard constraints and thus hard constraints should be avoided to the largest extent possible.)

Last, consider the constraint that *Pack and ship* should be allocated *round robin* (i.e. by rotation) to employees with the *Inventory* role. Here we are dealing with a constraint that is much less significant than any of the others. It reflects a goal that employees should get a feeling of fairness by carrying out their task in a turn-based fashion, but it is also clear that this constraint must yield to more important operational goals such as delivery time – at least temporarily. Again we have a constraint that should not be elevated to law, but adds guiding information to the process to help users reach soft goals. If we find ourselves breaking the rule systematically, then there is an incongruence between the prescribed process and the real-world process worth looking into. We may find that the inventory department is overburdened, but we are much more likely to find this if the rule is in the process description so that we can systematically record and diagnose violations. If the system does not let users break that rule, they will likely break it anyway, but in roundabout ways that are more difficult to monitor quantitatively or even detect.

Contributions and Outline

Our main contribution is to show how using soft constraints *directly in the business process descriptions* can (a) capture intentional information about best practices, (b) provide support to the user in extremely flexible BPMS, (c) help identify gaps between actual and prescribed practice, and (d) make the trade-off between flexible and rigidity significantly less salient. The paper presents:

- An analysis of the trade-off between flexibility and control from the user's perspective (Section 2). Specifically we argue that the user perspective leads to fundamentally different requirements, and that users *will* improvise and build ad hoc systems if sufficient flexibility is not provided.
- Issues related to getting flexibility in current systems (Section 3). Current systems provide a plethora of ways to specify flexible workflows, but an important advantage of strict control seems to have been forgotten: that of *guiding* the user. We argue that current systems cannot both be extremely flexible and still provide sufficient guidance, because intentional information is too easily lost when systems become too flexible.
- The case for using soft constraints/goals as a remedy (Section 4). This section explains how soft constraints/goals can alleviate the problem identified in Section 3, by allowing system to be flexible, while capturing enough information to guide the user through the process. While completely flexible systems will let the user do anything, flexible systems *with soft constraints* will allow full flexibility *while supporting the user* in making choices that contribute towards organizational goals.
- How soft goals look from the user's point of view (Section 5). This section shows how soft goals can be used in a direct way as part of what the user sees when interacting with the BPMS.
- How process designers specify soft goals (Section 6). Here it is explained how to incorporate soft constraints as a direct part of the process specification.
- How soft goals can be incorporated into the BPMS by using multi-criteria optimization (Section 7).
- How this affects the process design methodology (Section 8). This sections contributes a tentative methodology for using soft constraints when designing, deploying, and modifying business processes.

The paper closes with a brief overview of related work (Section 9) and directions for future research (Section 10).

2 BACKGROUND: THE USER PERSPECTIVE

Typically, business process modeling is initiated by management as an effort to optimize business processes, as in the vision of business process reengineering (Hammer, 1990). The focus lies on documenting existing processes, describing to-be processes, and finally, when the BPMS is running, to monitor and analyze actual processes for the purpose of making them more effective. In this way, the feedback provided by the BPM tool is used to manage and control the work practice of the organisation.

From a technological point of view, basic concerns with regard to BPMS are maintenance and migration, architectures and infrastructures, standards, exception handling, programming models and notations. Typical problems are how to centralize control of execution, how to make the BPMS interact with a number of different systems (e.g. ERP systems, legacy systems, Internet services) and users, and how to make runtime changes possible.

This article takes on a different perspective – that of the use context of the BPMS. The user perspective gives rise to different questions than if flexibility of BPMS is seen from a managerial or technological point of view. Thus our point of departure is the simple question: what is the purpose of a BPMS from the user's perspective? One partial answer is that business process modelling is about reducing the space of choices for the user by identifying a valid and limited set of options for action. The model provides a way to navigate the space of possible actions, indicating what might be the next activity for the user to engage in.

When taking the perspective of the user a central challenge for the BPMS is how to fulfil the conflicting requirements for *flexibility* in work practice and *control* of the work practice (for a discussion of this problem, see e.g. Bernstein 2000).

One might be tempted to think that from the user's point of view, unlimited flexibility in the BPMS would be preferable. However, there are several reasons for introducing restrictions into the system:

- Conventions of practice are captured explicitly in an artifact (they are documented)
- The right way of doing things does not have to be constructed, or re-negotiated, every time
- Process models can serve as *coordination mechanisms* to share knowledge about the state of the work (Schmidt & Simone, 1996)
- The system can support the user in not violating business rules (e.g. regarding legal issues, regulations, policies)

On the other hand, if the system prevents its users from making local adjustments of the normal flow when appropriate, it is too rigid.

A widely documented observation within the CSCW field is the gap between the specified processes and the actual work practice:

- Users make (well justified) workarounds (see e.g. Kobayashi, 2005)
- They use the plans as maps for *situated action*, not as scripts for executing their work (Suchman, 1987)

One explanation of the gaps between the anticipated and actual use of the system is the general observation that IT systems are not always used the way the designer intended (see e.g. Robinson, 1993).

However, the kinds of (mis)uses mentioned above might also indicate design problems inherent to the BPMS. The question is whether the way BPMS are designed today make them a support or a barrier to the actual work practice: are they flexible enough, or too rigid?

One source of too much rigidity is the conflict between a technological perspective and a use perspective on the business processes. It is tempting for an IT expert to see the user as just another resource that can be invoked to execute a task: human and system activities are captured within the same model, and *orchestrated* by the same centralized technology, the workflow engine. In a notation like e.g. *WS-BPEL (Thatte, 2003)* no distinction is made between human and system activities in the process. A different approach is taken in *Windows Workflow Foundation* (Chapell, 2005), where human and system workflows are represented in two different notations (sequential and state-based). The choice to have two separate notations is based on the assumption that human and system workflows are fundamentally different in nature, especially when it comes to their requirements for flexibility: human work practices are much more likely to change, also in an ad-hoc manner, than system flows.

The question is then: if the workflow system is not assuming that the users act like programs, executing their tasks in a procedural manner that corresponds to the way the workflow was modeled, what would be the right way to design a flexible system where the users would still benefit from the *navigational support* a BPMS can provide?

3 THE CURRENT STATE OF BPMS

Before proceeding further it is appropriate to examine the current state of affairs.

A fair number of current systems provide flexibility facilities for their users: Cancellation allows processes or parts thereof to be cancelled, tasks and sub-processes may be specified as skipable, users can choose to redo certain parts and of course the user always can make a choice between the alternative paths specified by the workflow designer.

In addition some systems provide policies (i.e. local or global rules) that govern in detail what is permissible in terms of data dimensions such as employee, role, location, project, customer type, etc. Such policies take into account the current data in the process instance and behave accordingly.

However, the current systems are built on the basis of the same modelling paradigm, namely that of *binary classification of sequences of activities*.

The activity of modelling under this paradigm can be understood as classifying all possible sequences of process-related events into allowable sequences and disallowable sequences.

Consider the following business process type[1]:

$$a \rightarrow (b \; XOR \; c) \rightarrow d$$

In essence, this business process type makes a classification. It classifies all possible sequences of the activities *a*, *b*, *c*, and *d* into those that comply with the description and those that do not comply. The sequences { <*a,b,d*>, <*a,c,d*> } comply; all others do not.

The example demonstrates a key issue in current systems: there is only one classification. One cannot state e.g. that <*a,b,d*> preferable to <*a,c,d*> or that <*c,d*> is acceptable, although not encouraged. The process designer is forced to make the unpalatable decision of making all possible runs either acceptable or unacceptable. Such a decision is often impossible to make correctly at design-time, and the result is then processes that are too restrictive or too lenient. Hence a more refined type of *specification* is needed, but merely inventing a clever specification in insufficient. The change from binary classification to finer classification must permeate the entire design/use methodology for business processes. See Section 8 for more on the methodology.

It is useful to consider the distinction between binary and finer classification from yet a perspective. When charting business processes, a business analyst is likely to arrive at a semi-formal description that contains many different *types* of constraints and goals. Goals and constraints come from a variety of sources (Regev and Wegmann, 2005) and therefore vary in significance. E.g. the analyst could have established that for three activities *a*,*b*, and *c* the sequence <*a,b,c*> is the best practice, <*b,a,c*> and <*a,c*> are acceptable, <*b,c*> can be done in exceptional cases, and other sequences are logically inconsistent. In other words, the analyst has captured *intentional information* about best practices along with physical constraints.

Yet today's BPMS demands that the process designer discard all the intentional information gathered by the analyst and specify the process as a control flow. If the designer writes

$$(a \; OR \; b) \rightarrow c$$

then the description has lost all intentional information and every part of the specification appears to be a physical constraint. Information about the best practice is lost as well.

This situation is clearly undesirable. The knowledge gathered by the analyst is lost because the system has no way of capturing it. The designer then specifies a process that reveals nothing about the nature of its dependencies and constraints, and finally the user (i.e. the knowledge worker) works with a system that forces one particular view of the process and considers all violations equally unacceptable.

Could one imagine a completely free system that would only *inform* the user about the process, but not force it? Let's consider two extremes of flexibility. The *dictatorial* process specification is one which allows only the strictly complying sequences and blocks the user from doing anything else. The *anarchistic* process specification is one which allows all possible sequences of the three activities *a*, *b*, and *c*. Such a specification provides maximum flexibility, but it must be based on the assumption that users know what they are doing (see e.g. Bider, 2005). Since it is not specified what sequences are preferable to others, the users will not get any support from the anarchistic specification.

To summarize, a *dictatorial specification* (rigid):

- Allows only what is narrowly prescribed by the process designer.
- Represents only anticipated sequences.
- Cannot accommodate deviations and exceptional cases unless they have been meticulously accounted for in the process description. If they have not, users are forced to invent workarounds, and valuable historic information is then not captured in the system. If deviations happen often, users will tend to systematize these in auxiliary, ad hoc systems. This leads to data fragmentation and lack of belief in the system.
- Does not represent how business is actually done, only how it should be done according to a process designer's incomplete view of the world.

An *anarchistic specification* (flexible):

- Allows every possible combination and repetition of known tasks.
- Does not represent or endorse any particular sequences.
- Provides no support to the user (cannot suggest what tasks to do in what order to reach the goal).

Neither of these extreme approaches is ideal seen from the user's perspective. In the dictatorial specification the user is supported in anticipated sequences, but denied any deviation. In the anarchistic specification the user is allowed any deviation, but supported in none. In *both* of the specifications and in *any trade-off in between* there is no ranking of different sequences. There is no way of saying what sequences are preferred, discouraged, etc. We can design rigid and flexible processes, but they remain simple binary classifications. Some sequences are acceptable, some are not. The dictatorial specification allows fewer, the flexible specification allows more, but they cannot be ranked according to preference. This is the limitation of the binary classification that most of today's BPMS employ.

What would be nice would be a combination: to be able to specify what sequences were preferred anticipated, but not have to make that specification a hard constraint. Some parts of the specification would be violable, others not. What is needed is a way of specifying this, without just allowing everything to happen indiscriminately.

Our solution is therefore to capture finer classifications of processes explicitly, or put differently, try to make the

[1] This description omits all other perspectives than the purely process-structural perspective, but it is sufficient for the following discussion.

intentional information gathered by analysts *explicit* in the process specification. This way the user can be supported while we retain the distinction between strict and soft rules. The following section discusses one way of achieving this.

4 MODELING SOFT GOALS

As mentioned in the beginning of Section 3 BPMS provide powerful capabilities for specifying permissible and non-permissible courses of action, but this remains a *binary classification* – either a policy is satisfied or not. In the current systems there has been a strong focus on extending expressiveness so that we can accurately specify *hard constraints*, but overlooking the equally important question of *soft goals*. Soft constraints are constraints that should be used as guidelines, but they can be violated in exceptional situations or if the user has a good reason. Soft goals (Yu and Mylopoulos, 1994; Soffer, 2005) are business objectives that are not *directly* linked to business output or performance. Soft constraints are operational guidelines promoting good performance on soft goals. Whereas hard constraints are usually very stable over the evolution of a process, soft constraints change more often.

As explained in Section 3, when users and designers are forced to work with systems that focus unilaterally on hard constraints, they face two equally displeasing alternatives: (1) Important soft constraints are inadvertently promoted to hard constraints because designers add them as policies. The result is a system that is too rigid and inspires hacks that circumvent the constraints by going outside the system. (2) Soft constraints are left out of the system altogether, meaning that important organizational goals are disconnected from, arguably, the organization's most important process tool.

If we accept the premise that BPMS need to account for soft constraints and/or soft goals in addition to hard constraints, the pressing question is this: how is this presented to the direct users of BPMS? We believe that making soft goals and soft constraints visible for the end users is a key issue.

The challenge when modelling both hard and soft constraints is to get the right balance between the two. BPMS must offer enough rigidity to satisfy hard constraints, whether they are business requirements, nonnegotiable logical constraints, data constraints, or externally imposed constraints, such as legislation (Soffer, 2005). At the same time, users must be afforded the maximal flexibility possible, but continually be informed about the impact of their choices on various organizational goals.

As mentioned above, it is not enough to simply add soft goal support to the BPMS. A change of methodology is required to ensure that soft goals are used through the entire process lifecycle – by analysts, designers, reengineers, users, and process miners. Our main focus here is the user perspective, and secondarily the designer's perspective; a more comprehensive methodology will be addressed in future work. For previous work on methodologies incorporating soft goals cf. (Yu and Mylopoulos, 1994; Regev and Wegmann, 2005). In the following we examine soft goals from the perspective of the user and the designer.

5 HOW USERS INTERACT WITH SOFT GOALS

In our example process in Figure 1 users will be presented to the activities as they become enabled, e.g. when a sales person is done entering an order received by telephone, the *Prepare and send invoice* activity becomes available to employees with the *Accounting* role, and the *Classify order and customer* activity becomes available to the sales person who entered the order.

In the following let us focus on two soft goals: minimizing *order processing time* and minimizing *credit risk*. Also assume that an automatic credit check takes up to five hours and a manual credit check takes two days. We can measure credit risk as zero when no policy violations take place and as the number of dollars in excess of the limit, when violations do take place. If a sales person, pre-approves a $1400 order, he has traded a *processing time* improvement of five hours with an added *credit risk* of $400. (The exact definition of such soft goals can be debated, and certainly more research on best practices in this field is required. We postpone this discussion to future work and accept the sketched goals for the purpose of illustrating a point.) The user interface must make this trade-off clear before the user finalizes his choice. The user, in other words, should be supported in navigating the decision space.

In general when the user is presented with a choice, the impact of each possibility should be explained in terms of soft goals. Some options are much less reasonable than others and naturally the user interface prioritize. For instance, performing a manual credit check on a $500 order adds nothing to the *credit risk* goal (perhaps a bit unreasonably) and delays the order by two days. This can be determined by a BPMS and supported in the user interface. This idea can be brought to bear on the notion of *dominant choices* introduced in Section 6 and 7.

As an alternative to optimizing and presenting solution pertaining to several goals, we can combine all soft goals into a weighted sum. While this simplifies the presentation – choice can simply be sorted based on one number – it has the drawback of maintaining a static priority between the goals. The weighting can be adjusted over time based on decision analysis, but despite its simplicity, we consider a weighed sum a supplement rather than a substitute for presenting the impact for each soft goal separately.

6 HOW DESIGNERS SPECIFY SOFT GOALS

Soft goals have a large variation in scope: some goals pertain only to one particular activity (the round robin allocation guideline in our example being an case in point), others pertain to part of a process (*credit risk*), and some may be influenced by many running processes of different

types (e.g. *processing time*). This indicates that soft goals may be independent of any particular process, but could also part of one. Hence a BPMS *must* allow goals both a system-level and at process-level.

Conceptually we can think of the change of approach in the following way: earlier, the designer would specify a process where every part of the specification was strictly non-violable; now, the designer will specify a process consisting of both a violable part and a non-violable part. The non-violable part will be much smaller than before, and the violable parts of the specification will be annotated with information about their impact on the soft goals identified by the process analyst. (We refer the reader to (Yu and Mylopoulos, 1994) for a method for representing the impact of activities on soft goals.)

Abstractly, we can think of a business process specification as simply a set of constraints and rules. Where previously every constraint or rule was strict, every constraint or rule is now either strict or soft. In the latter case it will annotated with enough information to automatically calculate its impact on soft goals at runtime.

It should be made clear that when we speak of a business process comprising constraints and rules, this includes everything in a process specification: control flow, dataflow, user allocation rules, exceptions, compensation, timeouts, etc. We should note that in most processes there will still be strict rules. E.g. data flows that feed input to computer activities would typically be non-violable, because the computer activity would otherwise fail and leave the process in an undefined state. We advocate to only use strict rules when they are categorically mandated as is the case with the data flow here. Other constraints can be expressed in terms of soft goals and result in a much more pleasing system to work with for all parties involved.

Let us examine several ways of specifying the violable parts (soft constraints) of the process.

(1) Goals can be specified directly as a function on the trace (i.e. history) of the process. E.g. *credit risk* would be specified globally as the sum of all violations:

$$z_{\text{Risk}} = \sum_{\text{traces}} \max([\mathit{Credit\_check} = \text{no}] \cdot (\mathit{total} - \mathit{autoLimit}), 0) + \sum_{\text{traces}} \max([\mathit{Credit\_check} = \text{auto}] \cdot (\mathit{total} - \mathit{manLimit}), 0)$$

While such a direct expression of the goal will be appealing to optimization adherents, its merits as a tool for process designers are debatable.

(2) A more natural, albeit indirect, approach is to annotate the activities in question locally with a measure of the gravity of a violation, e.g.:

$$\mathit{Total} < \$1000 \text{ OTHERWISE: } z_{\text{Risk}} \mathrel{+}= \mathit{Total} - \$1000$$

which states that for the particular path *Total* should be less than \$1000, otherwise the soft goal z_{Risk} is compromised by the *Total* - \$1000. In essence, rules that can be violated carry a price tag for a violation on them. The BPMS still needs to consider a decision horizon beyond just the current task, though, because more serious ramifications of the decision could be waiting further down the line.

Our example process has two soft goals, z_{Risk} and z_{Time}. With the augmented process specification as outlined above, the BPMS is able to calculate for each proposed step the immediate impact on the soft goals.

The specification methods outlined above are just two ways of specifying soft goals. Given the both local and global nature of soft goals, it is prudent to allow specification of goals on both levels. Some subgoals could be computed locally and then be composed to form the main goal globally. This approach retains some modularity and allows the code to be associated to its most closely related level of abstraction.

7 SOFT GOALS AS AN OPTIMIZATION MODEL

The field of operations management/optimization has been used widely in particularly complex resource allocation scenarios. Given the setup described here, optimization can be used automatically to aid users in their decision-making in a business process based on the process data and the soft goals and hard constraints captured in the process specification. Whereas formulating and solving optimization problems are typically fairly complex endeavours that do not apply directly to the soft goals, it is useful to understand the workings of the BPMS in terms of an optimization problem.

Our example has soft goals z_{Risk} and z_{Time}. In the simplest setup the BPMS would limit its horizon to only one step in the process. Hence for each enabled task, it would compute its impact on the soft goals, if it were to be executed. This is a straight-forward computation, whose result can be shown to the user to support decision-making.

This approach will often be insufficient because a locally attractive decision may lead down a path that adversely affects the soft goals. Hence a longer look-ahead is necessary, and this can be accomplished with an optimization problem formulation. It should be pointed out that the formulation will be seen neither by designers nor users; it merely defines how the BPMS finds optimal ways to progress so it can support the user by suggesting these. The BPMS builds the formulation based on the soft goal information provided by the designer.

Given a set of soft goals $z_1, z_2, \ldots, z_n$, the optimization problem can be rendered schematically as:

$$\begin{array}{ll} \text{minimize} & z_1, z_2, \ldots, z_n \\ \text{subject to} & \langle \textit{ hard constraints in process } \rangle \end{array}$$

where the decision variables describe the space of choices theoretically available to the user, and the hard constraints are derived directly from the process specification.

A concrete example will convey the intuition more clearly: consider the subset of our general order handling process shown in Figure 2. The decision variable has been made explicit in Figure 2 as *Check*, and it can take on one of

the values {*no*, *auto*, *manual*}. As before, there are two soft goals to minimize, namely *credit risk* and *processing time* defined as:

credit risk =
[*Credit_check* = no] · max(*total* – *autoLimit*,0) +
[*Credit_check* = auto] · max(*total* – *manLimit*,0)

proc.time =
[*Credit_check* = auto] · 5 hours +
[*Credit_check* = manual] · 2 days

where [*condition*] evaluates to 1 when true and 0 when false. The problem thus becomes

minimize *credit risk*, *proc.time*
subject to *Classify* before *Log*
Classify before *Check*
Classify before *Manual check*
Appr. = 1 ∨ *Pay* before *Pack*
(etc.)

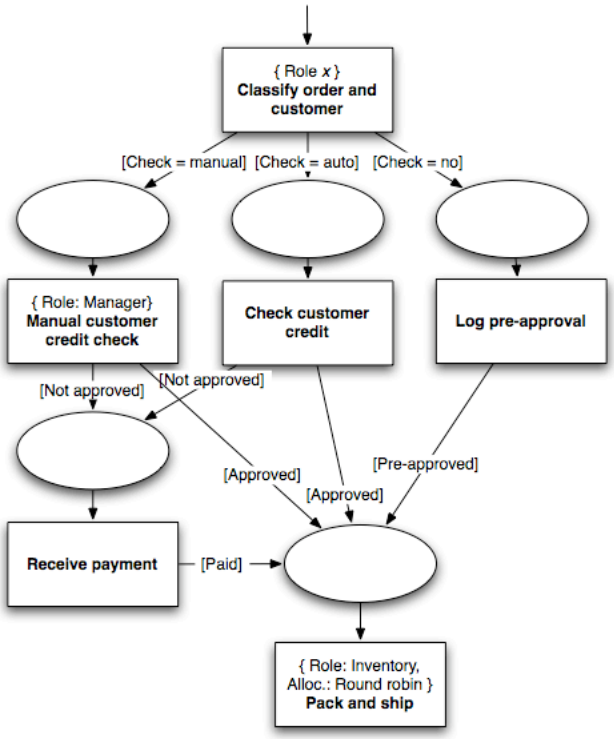

Figure 2 The credit approval part of the process

where *Act1* before *Act2* is a shorthand for introducing time variables *start* and *done* for the start and completion times of the activities, and demanding that *Act1.done* < *Act2.start*.

Again it should be pointed out that neither the user nor the designer will see this problem formulation. This is merely a conceptual way of describing the internal working of the BPMS necessary to support soft goals.

The optimization problem has several solutions, and in general there will be many more solutions than what can be easily presented to the user. To overcome this the system should present only *Pareto optimal* solutions (*dominant* solutions), i.e. solutions where no soft goal can be improved without adversely affecting another soft goal. Assuming we receive an order of $1400 in our example, at least three solutions are possible:

1. *Check* = no => *credit risk* = $400, *proc.time* = 0
2. *Check* = auto => *credit risk* = $0, *proc.time* = 5 h
3. *Check* = manual => *credit risk* = $0, *proc.time* = 2 d

Choice 3 is dominated by choice 2 (but not by choice 1!), and choice 1 and choice 2 are Pareto optimal and do not dominate each other. Hence choices 1 and 2 will be the first ones to be suggested to the user.

The system would also find a number of solution variations pertaining to the choice of what employee carries out the *Pack and ship* activity, but that decision variable is neutral to the decision at hand.

Note that by solving the optimization problem, we get a complete plan for action – not just for the current decision, but for the entire process. While this may prove unnecessary or too computationally intensive in practice, this is a convenient feature to be able to invoke occasionally.

Every time a step is taken, the optimization problem is rewritten to reflect the new state of the process. The optimization problem formulation should be derived automatically from the workflow specification and the soft goals, so the user or the designer will never see anything like the minimization problem above.

DISCUSSION

The ability to specify soft goals enables the BPMS to support the user in navigating the process. However, the optimization formulation relies on more than just soft goal specifications; it relies on accurate probability estimates of future event sequences. Hence, the conceptual model where an optimization problem formulation is extracted from the process specification is appropriate, but it is important to note that the size of its look-ahead depends sharply on the quality of event sequence probability distributions available. In routine cases these can be extracted from a process log, but most often compromise is necessary in the form of a reduced look-ahead in the optimization step.

8 TOWARDS A NEW PROCESS DESIGN METHODOLOGY

The intent of this paper is to introduce soft goals as a feature n BPMS, but the intent is also to engender a revision of the method by which business processes are designed. This paper cannot fit a comprehensive revised process design methodology, but an outline is in order – what remains is deferred to future work.

Rigid constraints are the most common source of workarounds in BPMS, and lack of support is the most common source of confusion. Soft goals promise to alleviate these shortcomings, but they must be used gratuitously – also in situations where one would be inclined to use a hard constraint.

As mentioned soft goals will only be successful when they are pervasive in the process life cycle – not just in the BPMS. The normal process life cycle is to (1) analyze the business (2) design/adapt the process, (3) use it, (4) look for improvements, (5) repeat.

The analyst should aim to describe and classify the dependencies between soft goals and activity, e.g. in the style of (Yu and Mylopoulos, 1994). The designer can then incorporate these into the process specification, and a BPMS that understands such a process specification can now provide better user guidance. In other words: the intentional business information must be retained throughout the lifecycle of the process specification.

In the evolution of the process, we strongly recommend to start out with the most liberal process specification possible: this means (a) prefer soft goals to hard constraints and (b) prefer parallelism to sequence. A similar idea to (b) has been mentioned by Bider (Bider, 2005). We have not left out the hard constraints altogether, instead we have recast them as soft goals; that is, we say "our goal is to violate the constraints as rarely and as insignificantly as possible" instead of "this constraint can never be violated". Whereas hard constraints are likely to lead to frustration, soft constraints allow us to capture intentional information in a non-obstructive way. A process specification stands a better chance of success if the designer errs on the side of flexibility rather than control. To add constraints to a flexible process one simply needs a documented reason. To remove constraints from a rigid process one must identify workarounds and diagnose them, and one can never be sure to have found them all. Situations that require new workarounds may occur at inconvenient times in the future. Removing an unnecessary constraint can take time causing delays to the process instances in question

This is not to say that identifying soft goals is trivial. There may be widely disparate opinions among the stakeholders of a process, they may express goals imprecisely, or there may simply be too many goals to realistically capture. While this is very likely to be the case, it may in fact increase the usefulness of a system that is able to capture soft goals: it forces users to reflect upon and discuss soft goals, and – if applied optimally – enables the discussion of soft goals to happen continuously. Upon explicitly representing soft goals in the system, users may find that the goals have become more narrow in their scope that the original formulation (to give an extreme example the expressed soft goal "ensure worker comfort" becomes the very narrow soft goal "schedule job A by round robin when possible" in the system). This demonstrates that soft goals – exactly like the processes they are part of – should evolve continuously and with help of their users. More research is needed to identify best methodological practices in this area.

9 RELATED RESEARCH

The topic of offering more flexibility in BPMS has recently been treated extensively (Schmidt, 2005; Bider, 2005; Soffer, 2005; Borch and Stefansen, 2006). Soffer introduced a taxonomy between hard and soft constraints, and applied these labels to general categories of constraints such as environmental constraints, sharing dependency constraints, goal reachability constraints etc. (Soffer, 2005)

In a recent paper Regev, Bider, and Wegmann presented an approach to flexibility using invariants (Regev, Bider, Wegmann, 2007). Conceptually a process run is seen as a trajectory through a state space and time, and process specification then becomes a matter of limiting the trajectory while finding allowable movement in space-time that brings the process closer to the preset goals. In terms of that view of the world, out work here can be seen as systematically ranking sets of states and trajectories as *preferred*, *neutral*, *discouraged*, etc. for each soft goal.

In other words, the work presented in this paper should be seen as an augmentation of existing systems, that is independent of – but compatible with – the view of the world presented in (Regev, Bider, Wegmann, 2007). The languages used in the BPMS can largely be left in place, and flexibility is gained through extra soft goal annotations.

A different approach to flexibility is that of Sadiq, Orlowska and Sadiq who allow *pockets of flexibility* inside typical strict workflows (Sadiq, Orlowska, Sadiq, 2004; Sadiq, Sadiq, Orlowska, 2001). A pocket of flexibility allows the users to build part of the workflow ad hoc inside the pocket, which is connected to the rest of the workflow as if it were an activity or a subworkflow. What goes inside such a "build" area is governed by a simple constraint language. While the motivations driving this work are virtually identical to the ones listed here, the outcomes are remarkably different. Sadiq, Orlowska and Sadiq propose to make the workflow *regionally* amenable to *construction at runtime*. Again, we would argue that while this certainly does provide more flexibility, it does not address the fundamental problem that not all sequences are equally good; a ranking (e.g. *preferred*, *neutral*, *discouraged*) expressed as a soft constraint is needed for each soft goal.

It is common to distinguish between changes to a process template (type-level) and changes to a running process (instance-level) (Regev et al., 2005). This paper is chiefly concerned with the type-level. We have not proposed a faster way to change process templates, nor have we proposed a method for changing running processes; we proposed adding soft goals and soft constraints to the process specification language, so that we will need to change both templates and instances less frequently and converge faster to a useful, but flexible, process description.

Very recently a similar approach was proposed leveraging an optimization engine for solving allocation issues on-the-

fly (Hamadi and Quimper, 2006). The approach presented here however is more comprehensive in that it is ubiquitous in the process and its description and not constrained to allocation.

Last, it is important to point out the large body of work on *soft systems*. The application of soft constraints and soft goals in BPMS can be seen as an instance of the general *Soft Systems Methodology* (SSM), a comprehensive methodology dealing with situations where people have diverging – or even contradicting – views on problem definitions and goals. For more on SSM, see e.g. (Checkland and Scholes, 1990).

10 CONCLUSION AND FUTURE WORK

Current systems and theories allow designers to specify binary classifications of processes. We have argued that this is insufficient, and that it inevitably leads to rigid (dictatorial) processes or support-less (anarchistic) processes. We have also argued that the flexibility given to the user by other approaches, while useful, lacks the explicit representation of how any future course of action affects the set of soft goals. Clever specifications can only partially remedy this, because the classification remains binary given the current technology.

Soft goals seek to alleviate these problems. They allow designers to specify goals instead of rigid constraints, and allow users to immediately see the gravity of their violations. We believe that this allows systems to be very flexible while retaining the support found in systems with an emphasis on control.

To deliver on these promises, soft goals must be made explicit and be understood by the BPMS. We have outlined how this is done, and proposed the necessary changes to the design methodology employed by process designers.

Processes evolve in response to ideas and to changes in the world. It presupposed here that process designers will write goals themselves based on managerial targets. In the future this is more likely to be a symbiosis between managerial targets and inspiration fed back to the designers via process mining. To keep the goals up to date the process logs should be constantly mined for violations, new constraints etc. so that they can be incorporated into the process repository.

We have postponed a treatment of data-dependencies. Data-dependencies to a large extent (but not entirely) are hard constraints, but they can be used to derive sequential constraints. If a system derives such dependencies itself, the designer has one design burden less, and any changes in the data will immediately be reflected in the appropriate constraints and nowhere else. On the other hand data-dependencies also enforce sequential constraint and any flexibility afforded to the user, must ensure that data-dependencies cannot be violated. Formalizing this remains a challenge for future work.

On a more general note it remains to be investigated how dependency-driven and state-based processes interact with the ideas presented in the paper.

REFERENCES

Bernstein, A. (2000), 'How can Cooperate Work Tools Support Dynamic Group Processes? Bridging the Specificity Frontier', *Proceedings of CSCW'00*, *Philadelphia.*

Bider, I. (2005) 'Masking flexibility behind rigidity: Notes on how much flexibility people are willing to cope with', *Extended abstract of keynote talk, BPMDS'05. Proceedings of the CAiSE'05 workshops, Vol. 1, FEUP, Porto, Portugal.*

Borch, S. E. and Stefansen, C., 'On Controlled Flexibility', *BPMDS'05. Proceedings of the CAiSE'05 workshops, Vol. 1, FEUP, Porto, Portugal.*

Chapell, D. (2005), 'Introducing Microsoft Windows Workflow Foundation: An Early Look', at http://msdn2.microsoft.com/en-us/library/aa480215.aspx71-92768-6, accessed May 1, 2007.

Checkland, P.B. and Scholes, J., (1990) 'Soft Systems Methodology in Action', ISBN 0-471-92768-6, *John Wiley & Sons Ltd.*

Hamadi, Y. and Quimper, C.-G. (2006), 'The Smart Workflow Foundation', *Microsoft Research, MSR-TR-2006-114*, November.

Hammer, M. (1990), 'Reengineering Work: Don't Automate, Obliterate', *Harvard Business Review, July/August*.

Kobayashi, M. (2005), 'Work coordination, workflow, and workarounds in a medical context', *CHI'05 extended abstracts on Human Factors in Computing Systems, Portland*.

Regev, G., Bider, I., and Wegmann, A. (2007), 'Defining Business Process Flexibility with the Help of Invariants', *Softw. Process Improve. Pract., Vol. 12, 65-79, Wiley Interscience*

Regev G., Soffer P., and Schmidt R., (2006), Taxonomy of Flexibility in Business Processes, Workshop on Business Process Modeling, Design and Support (BPMDS'06), Proceedings of CAiSE'06 Workshops, p. 90-93.

Robinson, M. (1993), 'Design for unanticipated use', *Proceedings of the European Conference on Computer-supported Cooperative Work, Milan*.

Sadiq, S. and Orlowska, A. and Sadiq, W. (2005), 'Specification and validation of process constraints for flexible workflows', *Information Systems*, 30 pp 349–378

Sadiq, S. and Sadiq, W. and Orlowska, A. (2001), 'Pockets of Flexibility in Workflow Specification', *ER '01: Proceedings of the 20th International Conference on Conceptual Modeling*, pp 513–526, Springer

Schmidt, R. (2005), 'Flexible Support of Inter-Organizational Business Processes Using Web Services', *BPMDS'05. Proceedings of the CAiSE'05 workshops, Vol. 1, FEUP, Porto, Portugal.*

Schmidt, K. and Simone, C. (1996): 'Coordination Mechanisms: Towards a Conceptual Foundation of CSCW Systems Design', *Computer Supported Cooperative Work, Vol. 5, no 2/3.*

Soffer, P. (2005) 'On the Notion of Flexibility in Business Processes', *Proceedings of the CAiSE'05 workshops, Vol. 1, FEUP, Porto, Portugal.*

Suchman, L.A. (1987), 'Plans and Situated Actions: The Problems of Human-Machine Communication', *Cambridge University Press*, December.

Thatte, S. and others (2003), 'Business Process Execution Language for Web Services, Version 1.1', at http://www-128.ibm.com/developerworks/library/specification/ws-bpel/, accessed May 1, 2007.

Yu, E. and Mylopoulos, J. (1994) 'Using Goals, Rules, and Methods to Support Reasoning in Business Process

Reengineering', *Proc. 27th Hawai'i Int'l Conf. System Sciences, Maui, Hawai'i, Vol. 4, pp 234-235*, January