On Controlled Flexibility

Signe Ellegaard Borch[1] Christian Stefansen[2]

[1] IT University of Copenhagen, Rued Langgaards Vej 7, 2300 København S, Denmark
elleborch@itu.dk
[2] DIKU, University of Copenhagen, Universitetsparken 1, 2100 København Ø, Denmark
cstef@diku.dk

Abstract.
Striking a balance between rigidity and flexibility is a central challenge in designing business processes. Striking this balance begins on the type level, because expressiveness on the type level is essential to control flexibility on the instance level. With this paper we would like to start a discussion on the ability to accurately *specify* flexibility on the process type level.
The paper proposes a more detailed categorization scheme than the common categorization of processes into allowed and disallowed sequences. Specifically, three ideas are put forth: (1) Use a finer granularity in classification, (2) make fine-grained classification possible across several dimensions, and (3) provide formalisms, design tools, and systems to make such classifications easy, adaptable, and *directly* supported in design tools and BPS systems.
In addition, the paper poses several open questions related to the suggested approach, most importantly: How can this finer classification scheme be built into current formalisms, tools, and systems and how can we make it pleasant for designers and users to interact with them?

Introduction

The activity of modeling at the business process type layer can be seen as a classification of all possible sequences of tasks into allowable sequences and disallowable sequences. When one uses a business process tool to classify sequences of tasks, the challenge is striking a balance between support and flexibility. In other words there are two dangers:

1. Too much is allowed
2. Too little is allowed

If too little is allowed, the user will find the system too restrictive because it disallows sensible ways of working through the process, and the user will repeatedly have to modify the process on the instance level. If too much is allowed, the system might suggest tasks that are not ready, and it may be difficult to use by inexperienced users, or it may introduce inconsistencies in data because some tasks were handled in the wrong order. In other words, if the system is too flexible, it will not *support* the users in doing their work.

However, the classification into simply allowed and disallowed sequences might not be fine grained enough: ideally, a business process management system should let designers express more complex relations between tasks, and support the end-users with the appropriate kind of guidance, depending on the context of use.

The objective of business process support (BPS) systems research should *not* be to make BPS systems as flexible as possible, as also pointed out in [1] – the goal should be to provide means of *controlling* flexibility to get the "right rigidity".

In the following we consider several issues in attaining such *controlled flexibility*. Section 2 discusses the problems of the simple *accept/deny* type of classification found in many systems, section 3 extends this problem to a setting with multiple inter-related dimensions, and section 4 discusses the issue of how designers and users might interact with such a richer system. Section 5 presents points for further discussion, and section 6 outlines future work, followed by a conclusion in section 7.

Balancing Support and Flexibility

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.

Let's consider the two extremes of flexibility. The *dictatorial* BPS system is one which allows only the two complying sequences and blocks the user from anything else. The *anarchistic* BPS system is one which allows all possible sequences of the four activities a, b, c, and d.

Neither of these extreme systems is ideal when it comes to satisfying the needs of an organization.

An anarchistic system could, for example, present an unstructured task list, where it is up to the user to decide in which order to do the tasks. Such a system would be based on the assumption that people know what they are doing, and that the system should support their work with friendly reminders, but interfere as little as possible, see e.g. [4]. However, some users might need more support than the anarchistic system provides. The right balance between support and flexibility in a system depends among other things on the level of education of the users: how well do they know their tasks, and how well do they know the system.

So far we have pointed out some of the problems pertaining to the classification of process flows into allowed and disallowed sequences.

It should be possible to **use a finer granularity in the classification**. Perhaps the categories could be *recommended*, *suggested*, *allowed*, *discouraged,* and *denied* (or some other classification appropriate to the context). Instead of classification into

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

allowed or disallowed sequences, we imagine a continuum, where absolute prohibition is at one end and optimality/best practice is at the other end. The designer specifies which sequences belong where in this spectrum.

Controlling Flexibility Across Several Dimensions

Any realistic BPS system is likely to support several *dimensions* in addition to the pure process description. Such dimensions might be *users, roles*, *time constraints, customers*, *market segments*, *locations* or *the employee's experience* to name a few. The key observation is that it is not enough to classify sequences of activities isolation; several dimensions play into the classifications.

Consider two examples: (1) Allowing certain activities to be skipped was a tremendous improvement to early-day systems, but a binary "can be skipped" flag on each task is unlikely to work well in practice. Dependent scenarios such as "can be skipped if less than two days left" or "can be skipped by managers" are much more likely. (2) Complicated relations like "managers and supervisors can carry out activity *b*, but managers are recommended, unless time left is less than four days in which case the supervisor who did activity *a* is preferred" simultaneously use several dimensions in the process description *and* a finer classification than a simple comply/not comply.

Describing such processes without over- or under-specification is immensely important, because – as stated before – neither a dictatorial nor an anarchistic system is desirable – certainly even less so with more dimensions involved.

We must be able to **classify relations between tasks along several dimensions**, e.g. classifying depending on parameters such as *user role*, *time constraints*, *customer*, or *the employee's experience*.

From Novice to Developer: Users' Rights to Make Changes

Another aspect of the use side of flexible systems is *who* is allowed to make changes. Skilled users tend to make workarounds when working with rigid systems [2]. One could argue that a flexible system should be designed to let educated users put their knowledge into the system, instead of making workarounds.

The system should allow users to skip tasks and restructure running processes, as well as making changes to the process definitions on the process type level, depending on the users' experience and authority. Novice users should be allowed to make only limited changes whereas experienced users with organizational responsibility should be able to make fundamental changes.

However, one should take the use context into account – the design of user access rights might differ substantially, e.g. depending on whether the system is designed for production or office work. In a production line, other restrictions exist, such as material ones, which means that users should not easily be able to change behavioral or operational aspects of the system.

With categories graduated from *allowed* to *denied*, flexibility becomes a question of expressiveness, namely: Can the business process designer *easily* and *accurately specify* which traces belong in which categories? If the process designer can do this, we have gained *controlled flexibility*. We must provide tools and formalisms for the process designer, or experienced user, to **make this classification as easy and flexible as possible**.

Questions to be discussed

We propose a discussion about the following questions:

1. What granularity is needed to classify sequences of activities? Are three categories (*recommend*, *accept*, *deny*) sufficient?
2. How can we add the notion of classification to the various process description dimensions in current tools and formalisms? A more concrete example: if the designer wishes to specify that activity *b* can be skipped, but only by a user with role *Manager* and only if the process time-to-finish is less than 4 days, how can such controlled flexibility (tying together several dimensions) best be accommodated?
3. The designer should be extremely prudent when classifying sequences in the *deny* category. Can we put forth a proposal for best design practices regarding what should go into the *deny* category?
4. How might the UI of the end-user be affected by a more detailed flow categorization scheme? E.g. several choices of tasks could be presented but alternative tasks change their status on the task list once they are no longer strictly required.
5. The process design tools should provide ways of recommending what constraints can be violated and what cannot based on e.g. data-flow dependency, resource sharing constraints etc. How should this be built into the design tools? And how can the idea of controlled flexibility be implemented in tools in a way that makes it easy to work with for the process designer? As an example the color or the weight of an edge in a process graph in the design tool could signify whether the particular sequence is recommended, allowed or denied. This is currently not part of what process notations are able to express: e.g. the arrows connecting tasks do not specify this.
6. Is more than one dimension needed? (Perhaps several different soft-goal-based metrics?)

Future Work

An obvious generalization to be addressed in future work is having several classifications corresponding to several business (soft-)goals. Our example here with classes *recommended*, *suggested*, *allowed*, *discouraged*, *denied* could pertain to a best practice, but other metrics could be employed and give rise to more (orthogonal) classifications.

Using Fine-Grained Classification in the Feedback Loop

Suppose we have three categories: *recommend*, *accept*, *deny*. The BPS system may write all process instances that were only accepted to a log and occasionally present these to the designer with suggestions for improvements. In this way the categorization serves to create a feedback loop to the designer, thus enabling continuous adaptation and improvement on the business process type level.

The BPS system might also use such a classification to monitor soft goals. One soft goal could be lead-time, and sequences that fall in the "accept" category perhaps have a longer lead-time than sequences in the "recommend" category. The BPS system can now help management monitor soft goal achievement by reporting the fraction of instances completing in the "recommend" category.

Introducing Finer Granularity in Existing Process Descriptions

Introducing a finer granularity in existing processes may involve a lot of quite cumbersome manual work. Hence, an enticing idea is to be able to *derive* whether a sequence is *recommended* or simply *allowed*.

A crude first-approximation is simply considering data-dependencies and globally declared rules or goals as – in Soffer's terminology [3] – *essential* constraints (violation is denied) and other constraints as *inessential* (violation is allowed, but not recommended).

Deriving a finer classification from pre-existing processes may in fact be useful in two settings: it makes the transition to a fine-grained system easier and it alleviates some of the burden of specification in daily work.

Conclusion

This paper discussed the notion of *controlled flexibility* and put forth three suggestions for improvement:

1. **Use a finer granularity in classification**, that is, rather than just having sequences classified as *allow* and *deny*, use a finer spectrum such as *recommended*, *suggested*, *allowed*, *discouraged*, *denied* (or any spectrum appropriate to the context).
2. **Make fine-grained classification possible across several dimensions**. The classification across several dimensions should also be classified on a finer spectrum.
3. **Provide formalisms, design tools and systems to make such classifications easy and adaptable**. Simple being able to observe after-the-fact that a process followed best practice is not sufficient. The classification should be

ubiquitous: when designers describe processes, when users execute processes, when experts monitor processes, etc. A finer granularity improves nothing if it is not visible to designers and users.

Finally, we mentioned several points of discussion. The most important being how to construct formalisms, tools, and systems that help us attain controlled flexibility without overwhelming us with verbosity.

References

[1] Bider, I.: *Masking flexibility behind rigidity: Notes on how much flexibility people are willing to cope with*, Extended Abstract of Keynote Talk, BPMDS'05. In Proceedings of the CaiSE'05 workshops, Vol. 1, FEUP, Porto, Portugal, 2005.
[2] Kammer, P. J.; Bocher, G. A.; Taylor, R. N.; Bergman, M.: *Techniques for Supporting Dynamic and Adaptive Workflow*. CSCW: The Journal of Collaborative Computing, vol. 9, 2000.
[3] Soffer, P.: *On the Notion of Flexibility in Business Processes*. In Proceedings of the CaiSE'05 workshops, Vol. 1, FEUP, Porto, Portugal, 2005.
[4]Wulf, M.; Gryczan, G.; Züllighoven, H.: *Process Patterns - Supporting Cooperative Work in the Tools & Materials Approach*. Information systems Research seminar In Scandinavia: IRIS 19; proceedings, Lökeberg, Sweden, 10 - 13 August, 1996. Bo Dahlbom et al. (eds.). - Gothenburg: Studies in Informatics, Report 8, 1996. S. 445 – 460, 1996.