Human and Automatic Modularizations of Process Models to Enhance their Comprehension

xx[a]

[a]*xx*

Abstract

Modularization is a widely advocated mechanism to manage a business process model's size and complexity. However, the widespread use of subprocesses in models does not rest on solid evidence for its benefits to enhance their comprehension, nor are the criteria clear how to identify subprocesses. In this paper, we describe an empirical investigation to test the effectiveness of using subprocesses in real-life process models. Our results suggest that subprocesses may foster the understanding of a complex business process model by their "information hiding" quality. Furthermore, we explored different categories of criteria that can be used to automatically derive process fragments that seem suitable to capture as subprocesses. From this exploration, approaches that consider the connectedness of subprocesses seem most attractive to pursue. This insight can be used to develop tool support for the modularization of business process models.

Keywords: Business Process Modeling, Modularity, Empirical test, Automated discovery

1. Introduction

In the design and production of complex technology, modularity is recognized as a key principle. For example, it has been argued that the computer industry has dramatically increased its rate of innovation by adopting modular design [1]. In contexts such as these, *modularity* is commonly interpreted as the design principle of having a complex system composed from smaller subsystems that can be managed independently, yet function together as a whole [2].

Modularization is also applied in business process models using *subprocesses*. Most popular process modeling techniques support this concept, e.g. UML Activity Diagrams [3], EPCs [4], BPMN [5], and YAWL [6]. Various advantages are attributed to the use of subprocesses in process models, in particular when they grow large. At build-time, subprocesses support a *modeling style* of stepwise task refinement, stimulate *reuse* of process models, and potentially *speed up* the (concurrent) development of the overall process model [7, 8]. At run-time, i.e. when a process model is enacted by an automated system, subprocesses allow

for *scaling* advantages: Each subprocess, for example, may be executed on a different workflow server [8]. Finally, when a process model is used to facilitate the understanding of complex business processes among various stakeholders, subprocesses are supposed to ease the *understanding* of the model [9, 10]. The latter advantage is particularly noteworthy, because in most business applications it *is* the primary purpose of a process model to act as a means of communication [11, 12]. This paper will be concerned with this particular advantage of using subprocess in process models, i.e. the enhancement of their comprehension by human readers.

It should be noted that the way in which modularity is currently utilized in modeling practice raises some questions about its actual benefits from the perspective of human comprehension. First of all, there are no objective criteria to establish the right level of granularity for a subprocess. Accordingly, there is no absolute guideline if a particular subprocess should be on level X or $X + 1$ in a model hierarchy [13]. Neither is there a unique way to modularize a process model [13]. As a consequence, modularity is often introduced in an ad-hoc fashion. Furthermore, there are clearly drawbacks when the process logic is fragmented across models. In particular, it "becomes confusing, less visible, and tracking [...] paths is tiring" [14] if a subprocess is decomposed in further subprocesses. The fact that the semantic check in ARIS Toolset mainly addresses consistency issues between events in the subprocess and around the refined function illustrates the seriousness of this problem. Finally, even if modularization is useful for maintenance purposes, by making it easier to understand which aspects must be changed, it is questionable whether advantages materialize in practice: Many organizations fail to keep their models up to date [15].

In this paper, our interest is with two research problems. The first problem is that solid indications are missing for benefits of modularization in process models, i.e. the use of subprocesses, to ease their interpretation. Our interest is to discover whether subprocesses can be useful to improve the *understandability* of real-life process models. For this issue, we will build on an empirical investigation of two complex process models from practice, both in modular and "flat" form, and their comprehension by a group of 28 experienced process modelers. The contribution of our work is to provide tangible support for the usefulness of subprocesses in process models. We also provide an insight into the underlying causes for this effect.

The second problem we address is the lack of dedicated approaches to support process modelers with modularizing a given process model into a group of related, understandable subprocesses. We explore three attractive directions for the automated discovery of subprocesses, apply them to a real-life and complex process model, and evaluate the results against the modularization that experienced process modelers provided for the same model. Our contribution in this respect consists of providing concrete indications for the further development of automated discovery algorithms.

In the presentation of our contributions, we will build on some of our earlier work [16]. In comparison with this publication, we significantly extended the presentation and discussion of the experiment that was conducted to investigate

the effect of subprocess usage and updated the review of related literature. Beyond that, the use and evaluation of the automated discovery algorithms that is included in the current paper is completely new.

Against this background, the structure of this paper is as follows. In the next section, we will give a broader background for the concept of modularity, in particular with respect to process modeling. In Section 3 we will present the setup of our empirical test along with its results and a discussion. Section 4 presents our proposals for automatic support for subprocess discovery with a corresponding evaluation. Section 5 compares our contribution to related research. Section 6 concludes the paper.

2. Theoretical Background

In this section we discuss the theoretical background of our research. In Section 2.1 we present the essential concepts related to modularity in conceptual modeling. Section 2.2 revisits contributions on the modularity of process models. Section 2.3 takes a cognitive research perspective on process model modularity, and derives hypotheses on its costs and benefits.

2.1. Modularity in System Design and Conceptual Modeling

Often, the terms *modularity*, *decomposability*, and *hierarchy* are used interchangeably. However, according to [2], a modular system is not automatically *decomposable* in the sense that the modules can be easily managed independently. After all, it is possible to break a system into modules whose workings remain highly interdependent with the internal workings of other modules. Furthermore, as Parnas points out in his seminal paper on "information hiding", a modular system is not necessarily *hierarchical* [17]. To clarify these notions, consider Figure 2. In this figure, three abstract modular designs can be seen. In

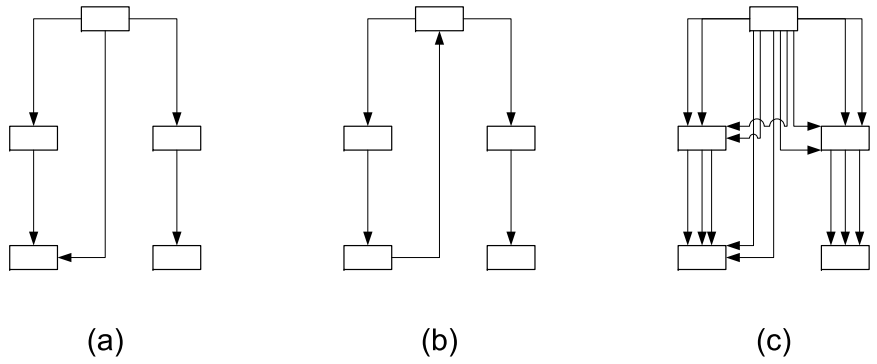

Figure 1: Examples of modular designs

each of these, a module is represented as a rectangle and each arrow represents a "uses" relation between two modules. Design (a) is hierarchical, since the

dependencies form a partial ordering. This is, however, not the case for design (b): A cyclic dependency exists between a subset of the modules. Such a design is called non-hierarchical. Furthermore, designs (a) and (b) may well be decomposable, considering the limited number of dependencies between the modules. In contrast, this is less obvious for design (c) with its numerous interdependencies. Note that the hierarchy notion can be mathematically pinned down, where decomposability refers to a qualitative notion. For this paper we consider the general phenomenon of "modularity" as the main subject of interest.

In many settings, "the real issue is normally not to be modular but *how* to be modular" [2]. Modular systems are much more difficult to design than comparable interconnected systems [1]. Beyond that, problems with incomplete or imperfect modularization tend to appear only when the modules come together and work poorly as an integrated whole. It has been argued that many of the most attractive and durable systems are developed through an "unselfconscious" design process [18]. In such a design process, used design rules are not explicit; inconsistencies and interdependencies are revealed by trial and error. However, it is by no means obvious that unselfconscious design must always, or even usually, result in modularity [2].

Quality criteria to consciously decompose a system into modules have been discussed by Wand and Weber on a general level [19, 20]. The authors identify five criteria. The first three are absolute criteria that are either met or not and focus on the content of the modular model, not its structure. *Minimality* requires that there is no redundant state information in the modular model. In data models this basically matches normalization requirements. *Determinism* requires that a state change is clearly identified to be triggered by an internal or an external event. If that is not the case the behavior of a module can only be understood by knowing the state of another subsystem. *Losslessness* demands that emergent properties are not lost in a modularization. Furthermore, the two criteria coupling and cohesion should be optimized, cf. [21]. *Coupling* should be minimal such that the sum of inputs of each subsystem is less or equal to the sum of inputs in any other modularization. *Cohesion* should be maximal such that all output affected by input variables are contained in the same set, and adding another output does not extend the set of input variables on which they depend.

Wand and Weber's criteria had a strong influence on the object-oriented design metrics proposed by Chidamber and Kemerer [22]. The usefulness of the five criteria has been demonstrated for UML class diagrams and state charts in an experimental setting [23].

2.2. Modularization in Process Models

The area of related research in the context of process models is huge covering works on *process modularization*, e.g. [24, 25, 26], *process inheritance*, e.g. [27, 28], and *reduction rules*, e.g. [29, 30, 31]. Since the latter two categories are mainly utilized for the purpose of process model analysis, i.e. the decomposition is non-persistent, we will focus on the first category. Furthermore, we do not consider modular design of process-aware information *systems* such as

in [32, 33]. In the context of process model modularization, three aspects can be distinguished: modularization operations, modularization prerequisites, and modularization selection.

Modularization Operations: The idea that *basic operators* should facilitate modularization was already proposed in the 1980s for data flow diagrams [24]. Refinement operations have also been defined for Workflow Nets [34]. Recently, the ability to extract a subprocess from a process model has been described as a change pattern for process-aware information systems [26]. This pattern must be implemented reflecting the syntactic requirements of the modeling language. In ARIS there are two ways to extract a subprocess: by modularization (refining function with subprocess) and by segmentation (cutting a model in different parts) [13]. Both these options are tailored to yield syntactically correct EPCs.

Modularization Prerequisites: There are some recommendations regarding the conditions *whether at all* a process model should be considered for modularization. Some of the practitioners books state that modularization should be introduced in a model with more than 5–15 [35] or 5–7 activities [10], yet without giving any support for this rule. Recently, it has been recommended based on empirical findings that process models with more than 50 elements should be decomposed [36]. Depending on the process modeling language the amount of activities can vary for 50 elements, e.g. EPCs use connectors for routing and events to separate functions while YAWL essentially only uses tasks. Still, up to now no objective criteria has been proposed for identifying which subprocess should be on which level in the model hierarchy [13].

Modularization Selection: There are some guidelines on how to *select* parts of process models for modularization. Good candidates for subprocesses are fragments of a model that are components with a single input and a single output control flow arc [37, 25, 38]. Furthermore, long and thin process models should be preferred to square models [13, p.278]. This argument points to the potential of metrics to guide the modularization. The idea here would be to use quality metrics like the ones proposed in [36, 39] to assess which modularization should be preferred. An application of metrics to compare design alternatives is reported in [40]. Yet, there is no dedicated approach to guide modularization based on metrics.

Overall, the main focus of research on process modularization is of a conceptual nature. Clearly, there are no objective and explicit guidelines that modelers in practice can rely on. The aim of our research as reported in the following sections is to contribute to a better understanding of the effects of modularization as a stepping stone towards such guidelines.

2.3. Cost and Benefit of Process Model Modularity

The modularity of a process model can have two major effects in terms of understanding: a benefit of *information hiding* and *browsing costs*. We discuss

them based on the *Cognitive Dimensions Framework*. This framework covers a set of aspects that have empirically been proven to be significant for the comprehension of computer programs and visual notations [41]. While the framework has been developed for notations, we can also use it to discuss comprehension of models in general or any other information artifact. There are two major findings that the framework builds upon: A representation always emphasizes a certain information at the expense of another one, and there has to be a fit between the mental task at hand and the notation [42, 43]. The implications of these insights for process models and their modularity can be discussed along the lines of those cognitive dimensions that are relevant for process model reading.

- *Abstraction Gradient* refers to the grouping capabilities of a notation. Most process modeling languages do not provide concepts for logically grouping activities from a single process model, although there are exceptions like BPEL [44], in which the scope concept can be used to logically group activities, and BPMN [45], in which the group concept can be used to logically group activities. In general, flow languages can be considered abstraction-hating [41]. As a consequence, the more complex the model gets the more difficult it becomes for the model reader to identify those parts that closely relate to one another.

- *Hidden Dependencies* refer to interdependencies that are not fully visible. In modular process models such hidden dependencies might exist between process model parts that are spread over different modules. This observation points to the potential danger that a more fragmented process model could imply a greater share of browsing costs, and therefore affect understanding.

Both these considerations result in the hypothesis that subprocesses are likely to increase a reader's understanding of a model due to information hiding. This assumption is measurable in terms of a suitable experimental design with understanding performance as the dependent dimension. Yet, it is up until now not clear to which degree additional costs in terms of browsing through different subprocesses might actually counter-balance the performance gain. We will discuss these issues in detail in the following section on our experimental design.

3. Experimental Design and Findings

This section presents an experimental design to test the effects of modularity (Section 3.1) along with its findings (Section 3.2) and a discussion (Section 3.3).

3.1. Research Design

In the previous sections we discussed that the ad-hoc way in which modularity is currently introduced in modeling practice raises doubts about its benefits, but that theoretical indications exist for the benefits of using subprocesses. In

developing a test for the presumed connection between modularity and understanding, several challenges must be met.

First of all, the question is how to pursue results that have the potential to provide insights that are meaningful, in the sense that they relate to the real-life application of subprocesses. For example, it would be unsatisfactory to test the effects of modularity in small or artificial process models. To achieve a realistic background for our research, we set up a collaboration with Pallas Athena Solutions[1] in the Netherlands, a specialized provider of BPM services. In our cooperation with this company, we gathered real-life, complex models as study objects. What is more, a large number of experienced process modelers from this company participated in our investigation.

The second issue relates to the organization of an empirical test in a rigorous manner. In lack of specific literature on empirical research with respect to process modeling, we build on approaches and classifications used in the field of software experimentation [46, 47]. In particular, we use an experimental design that is comparable to what was applied in a recent study to evaluate various types of BPM technology [48]. To test the hypothesis we carried out a so-called *single factor experiment*. In general, this design is suitable to investigate the effects of one *factor* on a common *response variable*. This design also allows to analyze variations of a factor: The *factor levels*. The *response variable* is determined when the participants of the experiment (who are also called subjects) apply the factor or factor levels to a particular *object*. The overall approach in our experiment is visualized in Figure 2. We will address each of the mentioned elements in our design in more detail now.

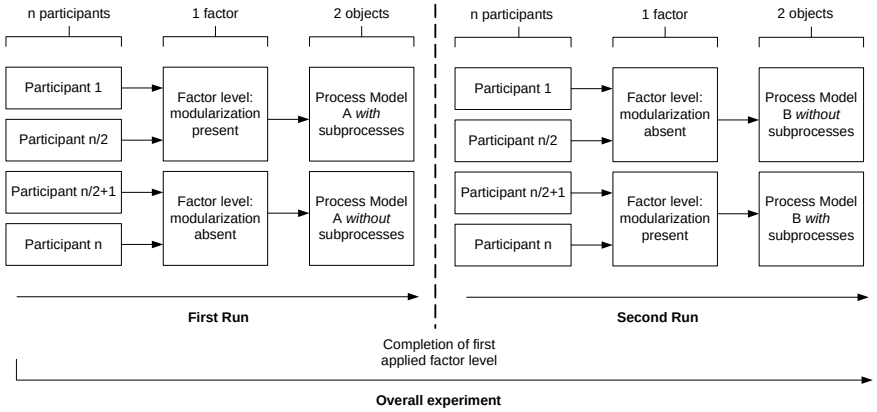

Figure 2: Experiment design

Object. The basic *objects* that were evaluated by the participants in our

[1] `http://www.pallas-athena.com`

test were two process models taken from practice. The models were used in the experiment both in their original form – displaying modularity – and in their "flattened version" where modularity is completely removed. The flattening involved the removal of all dependencies between model elements such that they all arrived at the same level of abstraction. Note that for any particular process model the absence or presence of modularity does not affect the business logic in a semantic sense.

The two process models were selected from a little over 80 process models that were created and delivered by our partner organization for its clients. We focused our search for suitable objects by the use of three criteria: (1) the presence of modularity in the process model, (2) the size of the model, and (3) access to the original creators of the model. The process models we looked for needed to display modularity as consciously applied by the modeler to deal with the complexity of a large model. We only considered models of more than 100 tasks, which can be considered as *very large* using the process size classification provided in [49]. Our line of reasoning here is that if modularity does not help to understand very large models, it will not help to distinctively understand smaller models either. Finally, we needed access to the modelers of the model to validate questions on the content of the model.

From our search, four candidate models emerged. One of these models was specifically developed for automated enactment and was not further considered. After all, the understandability of the model for human readers is generally not a prime issue in this context; the process model is automatically interpreted. Of the remaining three, which were all developed for the support of stakeholders in a process improvement project, the two process models were selected that were most similar to each other in terms of process size, number of subprocesses, and modularity depth. Both models had been modeled with the `Protos` tool [50], of which the underlying technique is similar to Workflow Nets [34]. The flattened versions of process models A and B can be seen in Figures 3 and 4 respectively.

Model A describes the procedure in use by an organization that is responsible for granting driver's licences. The process in question deals with clients that cannot directly obtain their driver's license because of physical or psychological disabilities that can influence their driving. Model B captures how a subcategory of unemployed citizens is coached and receives advice in finding a job. Note that the labels in Figures 3 and 4 have been removed to protect the confidentiality of the involved organizations; they were available to the participants in our test.[1]

Factor and factor levels. In our experiment, the *use of modularity* is the considered *factor*, with factor levels "present" and "absent". Note that we deliberately collected real process models from practice already exhibiting modularity and derived flattened versions from it, instead of doing it the other way around. In this way, we could build on a real-life application of modularity.

Response variable. The *response variable* in our experiment is the level of understanding that the respondents display with respect to the process models,

[1] A modular version of model A is available at `http://bit.ly/emYyTX`.

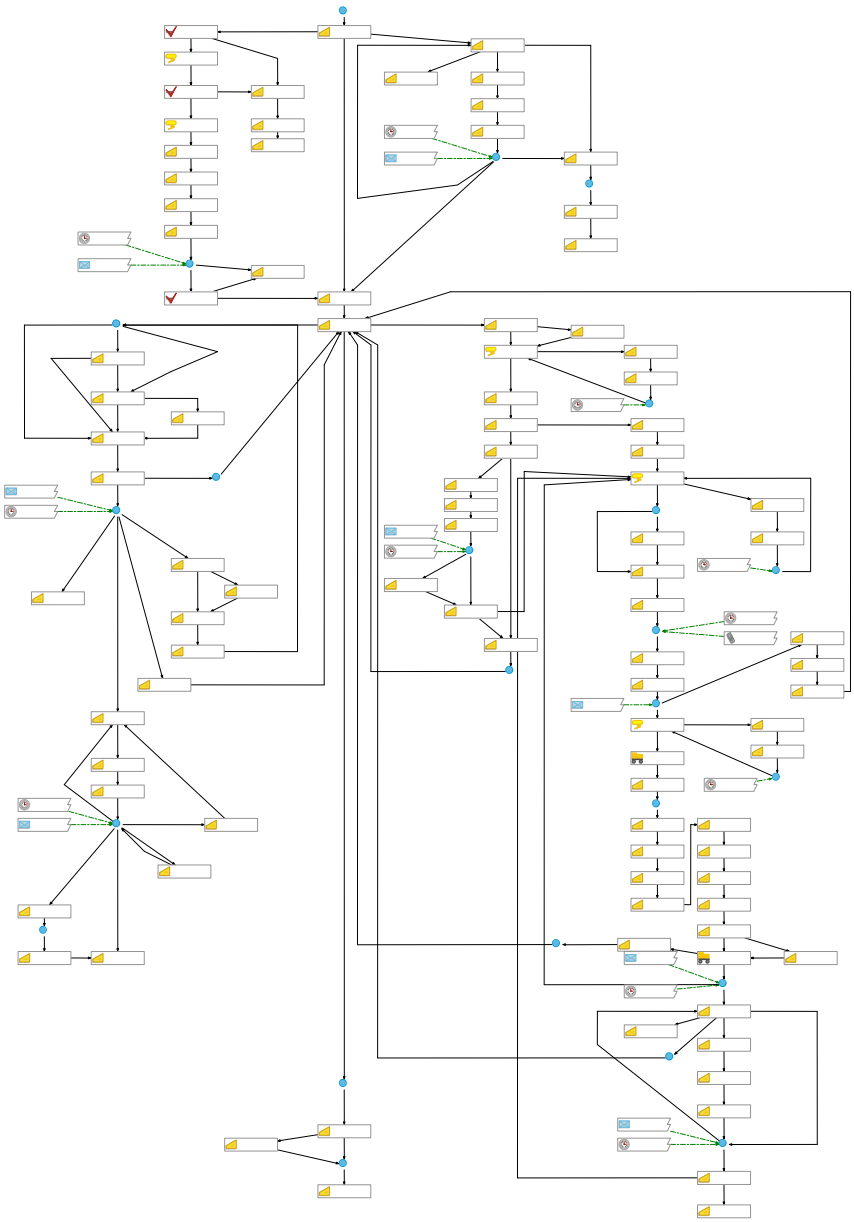

Figure 3: Flattened version of process model A

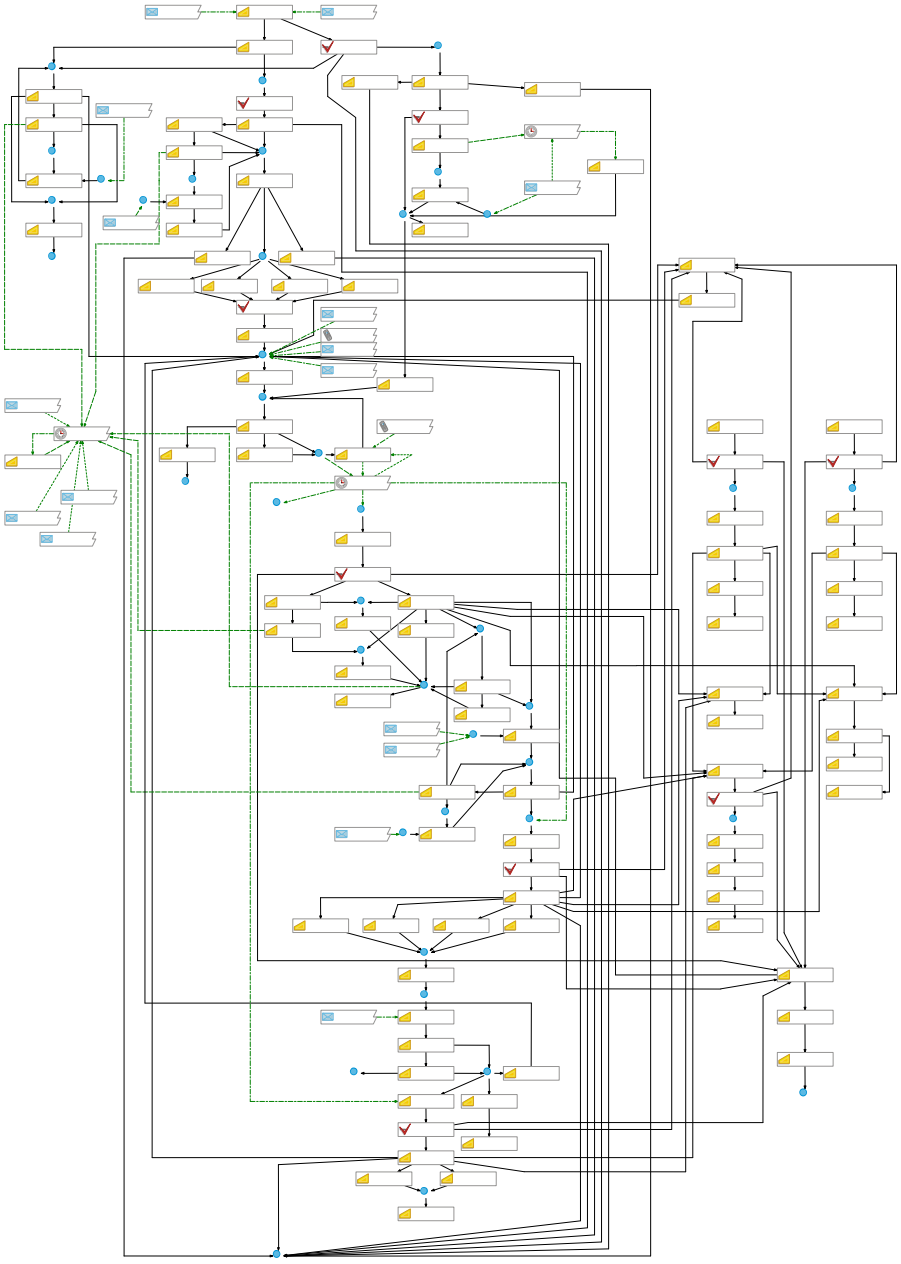

Figure 4: Flattened version of process model B

both in their modularized and flattened form. To measure the response variable, a specific set of questions was developed for each of the two models to be answered by the subjects. This approach is similar to the one we applied in a previous study into model understandability [39]. An example question for model A is: "If an AA-investigation is required, then a number of alternative settlements is possible. How many of these settlements exist?". For model B an example question is: "If a client does not appear on an appointment, is it always so that a new appointment is scheduled?". The questions were formulated in Dutch, the same language used by the creator of the modeler to name model elements, and also being the native language for all involved participants. The model-specific questions were preceded by a general introduction to the experiment, a specific explanation of each of the models, and a number of general questions with respect to the participants's background.

Subjects. The participants in this experiment were 28 experienced consultants from Pallas Athena. They were randomly assigned to the two groups used in our set-up (*block design*). Each group was presented two models: One model that displayed modularity and the other model in the flattened version. This way each participant received two *different processes* and two *different styles*. Participation in the experiment was voluntary; the single reward offered for participation was access to the research results.

Instrumentation. The participants were provided with the process models *on paper*, together with the questions; an alternative would have been to show the models on a computer display, e.g. using the software that was used to create the models. However, the involved company indicated that paper is the mostly used form to interact with their clients and that in the contexts in which the two models were used, this was also the case. Recall that the original versions of the models were divided into subprocesses by their respective authors. These models could therefore be presented to the respondents as a set of A4-sized papers, one for each subprocess. The alternative, flattened versions for both models were presented on A3 paper format, which allowed for reading the various labels with a normal effort.

Prior to the actual experimentation, all questions and correct answers were discussed with and approved by the creators of the models. They validated that the question sets could be used as a proper and representative way to test someone's understanding of the models. As a next step, five graduate students from Eindhoven University of Technology were involved in a pre-test. This led to the reformulation of 10 questions to remove ambiguities and the removal of 3 questions. The latter was explicitly required to ensure that the experiment could be carried out within a reasonable time frame. For both models, 12 questions were included in the final version of the experiment.

Data collection procedure. During the experiment, the subjects were asked to spend at most 25 minutes per model for answering its related questions. This limit was imposed to keep the time spent on the entire questionnaire under one hour and to prevent an imbalance in time spent on the two different models. Both at the start and at the end of answering a set of questions for each model, subjects were asked to write down the time to allow for time comparisons.

Table 1: Average percentages of correct answers for the model variants.

	Flattened	**Modular**
MODEL A	38.54%	42.36%
MODEL B	37.50%	58.33%

From the description of all the above elements it can be inferred that the experiment is *balanced*, which means that all factor levels are used by all participants of the experiment. In general, such an approach enables repeated measurements and the collection of more precise data: Every subject generates data for every treated factor level. As can be seen in Figure 2, we went through two runs, so this experiment displays a *repeated measurement*. However, in contrast to the approach in [48], two objects instead of one were used (process models A and B) to repeat the experiment in a second run. This setup enabled us to avoid the presentation of the same model content to the same group of subjects more than once, which limited learning effects.
In the next section, the results are presented of testing our hypothesis.

3.2. Results

For our data analysis, well-established statistical methods and standard metrics were applied, as provided by statistical software packages `STATGRAPHICS XV.II` and `SPSS 15.0`. In this section, we will first present our main analysis results, after which we will explore various alternative explanations for these to decide on our hypothesis.

Our main analysis focuses on the comparison between the group performance in terms of correctly answered questions for the modularized version and the flattened version of each of the models. In other words, does it matter whether someone sees a modularized or a flattened version of a process model? To determine the answer to this question, we calculated for each of the subjects the percentage of correct answers given for each model. Recall that each subject saw a modular model for one process and a flattened version of a model for the other process. The averages values are shown in Table 1.

As can be seen from this table, the modular version generates a *higher* average percentage of correct answers for both models, which suggests a better understandability. To determine whether the differences are statistically significant, it is important to select and apply the proper statistical test. Therefore, we first explored for each of the models the distribution of correct answers for each of its variants, i.e. the modular and flattened version. Because the standardized skewness and standardized kurtosis are within the range of -2 to +2, for each model the correctly answered questions can be assumed to be normally distributed. Additionally, F-tests indicated that with a 95% confidence the standard deviations of the samples for each of the models are also the same. These two conditions justify the application of Student's t-test [51].

Application of the t-test assuming a 95% confidence level results in the following results:

- For model A, there is *no significant difference* between the modular and the flattened version in terms of the average percentage of correctly answered questions (P=0.562).

- For model B, there is a *significant difference* between the modular and the flattened version in terms of the average percentage of correctly answered questions (P=0.001).

Even though these results – in particular the difference for model B – seem to support our hypothesis we must explore some alternative explanations to properly decide on its acceptance or rejection.

The most important *alternative* explanation for the differences between the results for model B is that – rather than whether the model is modular or not – differences between the experimental groups are the deeper cause. We analyze this argument in detail. Recall from Section 3 that our experiment is characterized by a *block design*, i.e. subjects are randomly assigned to the two experimental groups. If the subjects from the two groups were to differ in a characteristic feature that influences one's ability to understand process models, then this would offer a better explanation for the noted differences. A second, alternative explanation would be that the group of respondents that produced a higher average of correct answers for model B simply spent more time on answering the questions. After all, it is reasonable to expect that more answering time fosters a higher response quality.

To determine these alternative explanations, we analyzed the characteristics as shown in Table 2. Each entry in the table lists an investigated factor, the considered factor levels, and the P-value resulting from a statistical test. Note that we applied a standard t-test to determine a statistical difference between the groups with respect to each factor, unless its basic requirements were not met with respect to the underlying normal distribution and variance equality. If these requirements were not met, we used the non-parametric Mann-Whitney W test to compare the medians across both groups [51].

As can be seen from the P-values in this table, which are all greater than 0.05, none of the investigated factors signals a statistical difference between the groups at a 95% confidence level. Therefore, in lack of knowledge on other plausible influences, we conclude that *modularity appears to have a positive connection with process understanding.*

3.3. Discussion

For our discussion of the results presented in the previous section, we single out two questions:

- Why does modularity matter for understanding model B, but not for A?

- How exactly does modularity influence the understanding of model B?

We will address these one at a time, after which we will discuss the limitations of our experiment.

Table 2: Group comparison.

Factor	Factor levels	P-value
Domain knowledge	Knowledgeable with the process context or not	0.386
Company experience	Actual number of years within company	0.411
Field experience	Actual number of years working as process consultant	0.726
Education	University degree or not	0.453
Job type	Business consultant or technical consultant	1.000
Modeling amount	Estimated number of process models created	0.504
Modeling size	Estimated average size of process models created (nodes)	0.764
Time experiment	Actual time spent on entire experiment	0.948
Time B	Actual time spent on model B in the experiment	0.417

Model differences At this stage, we recall that we selected models A and B from a wide range of models by using a set of criteria (see Section 3.2). From the four models that met these, models A and B were *most* similar, notably with respect to the number of tasks they contain and their depth. To determine why modularity plays a bigger role in understanding model B, we carried out a further analysis of both models by using the metrics shown in Table 3. At the top of the table, some basic metrics are given, followed by metrics that have been proposed as indicators for process model complexity in general, and at the bottom some metrics that are explicitly proposed for assessing modular process models.

Two metrics display values that differ more than a factor 2 for the models under consideration, i.e. `Subprocesses` and `FanIn-Out`. According to [54], the relatively high value of the latter metric for model B (33.42) would suggest a poorer structuring of model B compared to model A. However, an additional test to determine whether a difference exists in model understandability between the modular version of model A and the modular version of model B does not show a higher average percentage of correct answers for the former. In lack of other empirical support for the use of this metric, the relatively high number of subprocesses (20) in model B seems more relevant: It suggests that the difference between the modular and flattened version of this model is more distinct than for model A.

For the remaining factors, models A and B display quite similar characteristics, even though model B is the slightly larger one. There is no general trend that suggests that one model is considerably more complex than the other and none of the metrics display large differences – other than the number of subprocesses. So, the most reasonable answer to the question why modularity has an impact on one model but not on another is that B's original version displayed

Table 3: Complexity metrics.

Metric	Description	Source	Model A	Model B
Tasks	Total number of tasks	–	105	120
Nodes	Total number of nodes	–	130	175
Arcs	Total number of arcs	–	171	248
Subprocesses	Total number of subprocess in original model	–	**9**	**20**
To	Average number of outgoing arcs from transitions (tasks)	[52]	0.81	1.03
Po	Average number of outgoing arcs from places (milestones)	[52]	3.42	2.24
Cycn	McCabe's cyclomatic number (adjusted for Petri nets)	[52]	43	75
Connectivity	Number of arcs divided by the number of nodes	[53]	1.32	1.42
Density	Number of arcs divided by the maximal number of arcs	[53]	0.020	0.016
AvgConDeg	Average number of input and output arcs per routing element	[53]	1.10	1.21
Fan-In	Average number of modules calling a module	[54]	1.25	2.26
Fan-Out	Average number of modules called by a module	[54]	1.5	2.26
FanIn-Out	$((\text{Fan-In}) * (\text{Fan-Out}))^2$	[54]	**3.63**	**33.42**
Depth	Degree of nesting within the process model	[53]	3	3

a much higher degree of modularization than model A. This may have helped subjects in understanding the model better. This would suggest that from a cognitive perspective, that model would show a better abstraction gradient and better information hiding.

The influence of modularity In search for an explanation of *how* modularity increases model understanding, we re-examined the questions we used in our experiment. Recall that these questions were validated by the original creators of the model (see Section 3): The questions were considered to be to the point and representative to test someone's understanding of the model.

In the post-hoc analysis of our results, we investigated the contention that by using a modular model certain types of question would be answered better than other ones. In particular, we categorized our questions as being of a *local* or *global* type. We characterized the difference such that an answer for a local question can be found within the confinements of a single subprocess in the modular version, where the examination of more subprocesses is required to answer a global question. As it turned out, model B contained 2 global questions and 10 local questions. In a comparison between the group that used the modular model and the group that used the flattened model, we found that for local questions, a *significant difference* exists in terms of the average percentage of correctly answered questions of model B (P=0.002). However, too few global questions were used to determine whether there is a difference in terms of the average percentage of correctly answered questions between using the modular or the flattened version of model B. Therefore, the cognitive effect of hidden dependencies do not show up distinctly.

From this analysis, we cautiously infer that modularity may be helpful for understanding a process model because it *shields the reader from unnecessary information.* Where the reader of a flattened model always sees the entire context, the reader of the modular version is confronted with a limited set of information when the proper subprocess is selected. This is especially helpful when such a reader is looking for local information. In this sense, the use of subpropcess in process models may generate an effect that is also achieved by Parnas' "information hiding" concept in software development[17]: Programmers are most effective if shielded from, rather than exposed to, the details of construction of system parts other than their own.

Whether there is also an opposite effect, i.e. that the correct answer for a global question can be easier found with a flattened model, could not be established for model B. Our suspicion is that this is not very likely; an analysis of the results for model A did not show such an effect.

Limitations Only 28 subjects were involved in this experiment and only 2 process models were considered. Both aspects are threats to the *internal validity* of this experiment, i.e. whether our claims about the measurements are correct. At the same time, these small numbers result from our choices to (1) involve real process modelers, as well as (2) process models from industrial practice. From all the modelers working in our partner company, more than half of them participated in the experiment and it was not feasible to involve them in a larger experiment, e.g. to have them consider more models. The choices

for real modelers and models clearly positively affect the *external validity* of our study, i.e. the potential to generalize our findings. Therefore, our experiment is another illustration of how "internal and external validity can be negatively related" [55].

Another aspect worth mentioning is the choice of displaying the process models *on paper*. It is by no means certain that findings that are similar to ours would result from an experimental set-up where models are shown on a *computer display*. Depending on the features of the modeling tool used, "information hiding" could also be achieved in other ways than applying modularity. For example, the `Protos` tool that was used to create the models also allows to zoom in on part of the model, while other tools may be able to print sub-graphs of process models or collapsed activity groups.

A third point for consideration is that the involved models were very large. It seems likely that the effects of the use of subprocesses in smaller process models are absent or at least less prominent. This is an observation that is probably comparable to the use of modularity in other technological domains.

Fourth, it is noteworthy that the process models without subprocesses that our subjects studied are flattened versions of once modular process models. If an experienced modeler were to capture the underlying processes without being allowed or being able to use subprocesses, we cannot rule out that the resulting process models would be different from these flattened versions. If this were true, it would be interesting to evaluate the understandability of such models in comparison with modularized versions thereof. At this stage we do not have access to large, industrial process models without any subprocess usage.

Finally, the lay-out of a process model may affect the understandability of a process model, as we hypothesized before [39]. As there is a limited understanding of such an effect for process models, we are restrained in properly controlling this variable. Note that we used the same modeling elements, the same top-down modeling direction, and a roughly similar breadth and width for both models to limit this effect (compare Figures 3 and 4).

4. Criteria for Subprocess Discovery

The results of the experiments we presented point to the usefulness of modularization to improve the understandability of a process model. An open question at this point is how to select parts of the process model for modularization? We referred to this issue in Section 2.2 as *modularization selection* and noted that no dedicated approaches for this issue are available. Clearly, a process analyst would benefit from a tool that assists in the discovery of subprocesses, i.e. collections of nodes that should be put together into a subprocess. As an explorative study towards the development of such a tool, we investigate criteria that are suitable for deciding whether nodes should be put together into a subprocess. We investigate three types of criteria:

- the block-structuredness of the subprocess,
- the connectedness of nodes in the subprocess, and

- the similarity of the labels of the nodes in the subprocess.

We explore the applicability of these criteria in a case study, in which we automatically divide model B from Figure 4 into subprocesses. The motivation to select this model is that it was developed by experienced process modelers and that their modularization showed tangible benefits for understanding it (see Section 3.2). We investigate both quantitatively and qualitatively how well the result from the automatic modularization approaches the subprocess division made by the human modelers in question. We first explain the three criteria in detail, next we explain the setup of an empirical evaluation of their appropriateness and finally, we discuss the results of the evaluation.

4.1. Block-Structuredness

A (sub-)process is called block-structured if it has a single *entry* and a single *exit*. For that reason, these blocks are often referred to as *Single Entry Single Exit* (SESE) components. The entry is the node through which the flow of control enters the block and an exit is the node through which the flow of control leaves the block. The requirement that a business process has a single entry and a single exit is quite common. For example, so-called block structured languages, such as BPEL [**?**], enforce this requirement by construction. It is also used as an additional requirement for other languages [34, 4]. Therefore, we consider this requirement a good basis for detecting subprocesses.

More precisely, let G be a directed graph with a single source and a single sink. A source is a node with outgoing arcs only and a sink is a node with incoming arcs only. Let F be a connected subgraph of G. A node is a *boundary node* of F if and only if it is either both connected to nodes in F and to nodes outside of F, or it if it is the source or the sink of the process. A boundary node is an *entry* if and only if all its incoming arcs are outside of F or all its outgoing arcs are inside of F. A boundary node is an *exit* if and only if all its incoming arcs are inside of F or all its outgoing arcs are outside of F. F is a *SESE component* if and only if it has a single entry and a single exit.

A SESE component is *canonical* if and only if its set of edges does not overlap with the set of edges of another SESE component. Since canonical SESE components do not overlap, they are either nested or disjoint. Therefore, we can form a tree structure of canonical components. The root of the tree is the entire process, the branches of a component in the tree are the components that are nested in that component. The resulting tree is called the Refined Process Structure Tree (RPST) [56, 57]. For this first approach of automatic modularization, we consider the RPST to form the subdivision of a process into subprocesses.

For example, Figure 5 shows a business process model that represents activities and control-flow relations between those activities. We abstract from the precise semantics of the notational elements. The process as a whole is a SESE component. Within the process, two SESE components can be recognized, one consisting of the vertices 3, 4, 5, 6 and 7 and one consisting of the vertices 9, 10, 11 and 12. Both of these components again consist of two components; the

first one consists of the components 3, 4, 7 and 3, 5, 6, 7; and the second one consists of the components 9, 10, 12 and 9, 11, 12. Hence, a tree structure is formed with the complete process as root, the components 3, 4, 5, 6, 7 and 9, 10, 11, 12 as branches and the sub-components of those components as further branches. Each of the SESE components can be marked as a potential sub-process. Figure 6 shows the RPST that is formed by these components.

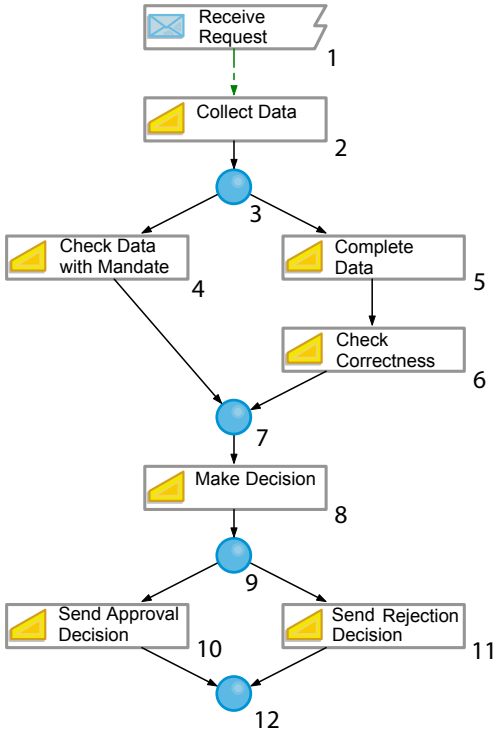

Figure 5: Example Process Model for Automatic Decomposition

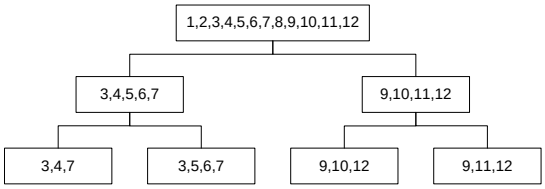

Figure 6: Example RPST

An algorithm for computing the RPST can be found in [56, 57].

4.2. Connectedness

A collection of nodes is connected if the nodes in the collection are more strongly connected by arcs to each other than to nodes outside this collection. Note that this definition implies that for an automatic discovery of subprocesses a parameter is involved that should express how strong the connection between nodes in a collection must be.

We use graph cluster analysis [58] to establish collections of nodes in a business process that are strongly connected to each other; in cluster analysis these collections are called *clusters* and they will be used as subprocesses. More precisely, we cluster a process as follows. Given a graph $G = (V, E, l)$ with vertices V, edges E and a function $l : V \to S$ that labels vertices and the number of clusters $n > 1$ that we aim to discover, our clustering technique partitions the set of vertices V into n clusters $C_1, C_2, \ldots, C_n$, such that the number of edges that cross clusters, defined as $|\{(n, m)|(n, m) \in E, n \in C_i, m \in C_j, i \neq j\}|$, is minimal.

For example, if we decide to identify 2 subprocesses in Figure 5, vertices 3, 4, 5, 6, 7 would never be put into separate clusters, because that would cause a cut of at least 2 edges (while the minimum-cut consists of only 1 edge). Similarly, 9, 10, 11, 12 would never be put into separate clusters. However, which two clusters are identified in the end is not deterministic, because as long as the nodes mentioned before are not put into separate clusters all other separations lead to a cut of 1 edge. For example, one possible separation distinguishes clusters 1, 2, 3, 4, 5, 6, 7 and 8, 9, 10, 11, 12. Another possible separation distinguishes clusters 1 and 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. These clusters can be marked as potential sub-processes.

Algorithm 1 shows the pseudo code of the algorithm that performs the clustering. It consists of four basic steps. In the first step (lines 3 to 10), a 'point' is created for each vertex. The set of points is the set Ps. Each point exists in a k-dimensional space, where each dimension corresponds to a vertex (and consequently k is the number of vertices in the graph). A point has the value 1 in a dimension in case the dimension represents the vertex itself or represents another vertex to or from which there is an edge. In the second step (line 11) an initial clustering is created at random. The set of clusters, Cs, is a partition of the points in the set Ps. The number of clusters equals the parameter n. In the third and fourth step, the set of clusters is improved, until it does not change anymore. In the third step (lines 13 to 16) the center of each cluster is computed. The center of a cluster is the point that has, for each dimension, the mean of the values that the cluster members have. CCs is the set of cluster centers. In the fourth step (lines 17 to 24), the set of clusters is then recomputed, by re-assigning each point to the cluster with the center that is closest to it.

4.3. Label Similarity

The third criterion we will use for the automatic modularization builds on the idea that nodes that have more similar labels can be considered to have a

Algorithm 1: Clustering Algorithm

input: $G = (V, E, l)$, n
begin
  $\mathsf{m} := \{1 \dots |V|\} \leftrightarrow V$
  for $i := 1$ *to* $|V|$ **do**
    $\mathsf{rv} := \mathbb{R}^{|V|}$
    for $j := 1$ *to* $|V|$ **do**
      $\mathsf{rv}_j := 1$, if $j = i$ or $(\mathsf{m}(i), \mathsf{m}(j)) \in E$ or $(\mathsf{m}(j), \mathsf{m}(i)) \in E$
    end
    $\mathsf{Ps} := \mathsf{Ps} \cup \{(\mathsf{m}(i), \mathsf{rv})\}$
  end
  $\mathsf{Cs} : \mathbb{P}(\mathsf{Ps})$, such that $\bigcup_{\mathsf{c} \in \mathsf{Cs}} \mathsf{c} = \mathsf{Ps}, \forall \mathsf{c}_1, \mathsf{c}_2 \in \mathsf{Cs} : \mathsf{c}_1 \cap \mathsf{c}_2 = \emptyset$, and $|\mathsf{Cs}| = n$
  repeat
    foreach $\mathsf{c} \in \mathsf{Cs}$ **do**
      for $i := 1$ *to* $|V|$ **do** $\mathsf{cc}_i := \frac{\Sigma_{(v, \mathsf{rv}) \in \mathsf{c}} \mathsf{rv}_i}{|\mathsf{c}|}$
      $\mathsf{CCs} := \mathsf{CCs} \cup \{\mathsf{cc}\}$
    end
    $\mathsf{Cs} := \emptyset$
    foreach $\mathsf{cc} \in \mathsf{CCs}$ **do**
      $\mathsf{c} := \emptyset$
      foreach $(v, \mathsf{rv}) \in \mathsf{Ps}$ **do**
        if $\neg \exists \mathsf{cc}' \in \mathsf{CCs} : d(\mathsf{cc}', \mathsf{rv}) < d(\mathsf{cc}, \mathsf{rv})$ **then** $\mathsf{c} := \mathsf{c} \cup (v, \mathsf{rv})$
      end
      $\mathsf{Cs} := \mathsf{Cs} \cup \{\mathsf{c}\}$
    end
  until Cs *is unchanged*
end

higher probability of belonging to the same subprocess than nodes that have very different labels. This idea is based on the assumption that subsprocesses can be developed to address a certain 'topic' in the process, like a phase or the processing of a particular information object, and that the node labels are chosen to reflect this topic. For example, it is more likely that the nodes labeled 'receive request' and 'judge request' belong to the same subprocess than the nodes 'receive request' and 'bill client'. In previous work [59, 60], we investigated several metrics to measure the similarity of two labels. In this paper we use the notion of *syntactical similarity*, which is based on string-edit distance. This is a rough metric that does not take complex relations between words, such as synonymy, into account. Its usage in this context can be defended because people who work on the same business process model can be expected to align their terminology and not use synonyms or homonyms. Consequently, we do not

expect more advanced metrics that take synonyms and homonyms into account to make a difference in performance. Metrics that consider 'relatedness' of labels can be expected to make a difference. We address such metrics in related work [61].

For clustering based on label similarity, control nodes are ignored, because their labels often do not provide information that can be used for clustering. For example, they can be labeled 'AND', 'OR' or 'XOR' to indicate their type. This does not provide us with information to relate them to other nodes in the process model.

The string edit distance is defined as follows. Let s and t be two strings. The string edit distance of s and t, denoted $\text{ed}(s,t)$, is the minimal number of atomic string operations needed to transform s into t or vice versa. The atomic string operations are: inserting a character, deleting a character or substituting a character for another. For example, the string edit distance between 'Verify invoice' and 'Verification invoice' is 7; substitute 'y' for 'i' and insert 'cation'.

Clustering can be used to determine clusters that are similar with respect to their labels in much the same way as it can be used to determine clusters that are well connected. This is done by minimizing the total similarity of labels between the clusters instead of the total number of edges between the clusters. More precisely, given a set of strings S, a preferred number of clusters n and a graph $G = (V, E, l)$ with vertices V, edges E and a function $l : V \to S$ that labels vertices, our label clustering technique partitions the set of vertices V into n clusters $C_1, C_2, \ldots, C_n$, such that the total similarity of vertices from different clusters, defined as $\Sigma_{n \in C_i, m \in C_j, i \neq j} \frac{1}{\text{ed}(l(n),l(m))}$, is minimal.

For example, using clustering to identify 2 clusters in Figure 5, forms the cluster 2, 4, 5, 6 (in which all nodes have 'data' and/or 'check' in their label) and the cluster 1, 8, 10, 11 (because 8, 10 and 11 all have 'send' and/or 'decision' in their label and 'receive request' is just most similar to 'send rejection decision'). These clusters can be marked as potential sub-processes.

Note that label similarity and connectedness can also be combined easily by only considering the similarity of labels in case the corresponding vertices are connected, i.e. we minimize: $\Sigma_{(n,m) \in E, n \in C_i, m \in C_j, i \neq j} \frac{1}{\text{ed}(l(n),l(m))}$. We can incorporate label similarity into the clustering algorithm, by assigning $rv_j = \frac{1}{\text{ed}(l(m(i)),l(m(j)))}$ in line 6. As will be discussed, we will also include this hybrid form of clustering in our evaluation.

For example, using a combination of label similarity and connectedness to identify 2 clusters in this case would consider the disconnectedness of nodes 1 and 11 and would produce the clusters 1, 2, 4, 5, 6 and 8, 10, 11. These clusters can be marked as potential sub-processes.

4.4. Evaluation Setup

We evaluated the applicability of the criteria by applying the criteria to automatically obtain a subprocess decomposition of the flattened process model B from Figure 4. We evaluated the decomposition both quantitatively and qualitatively.

For the quantitative analysis, we compared the following characteristics for the decomposition done by humans and the decomposition that was done automatically using the metrics: `Subprocesses`, `Fan-In`, `Fan-Out`, `FanIn-Out` and `Depth`, as they are defined in Table 3. The values for the other metrics do not differ for different decompositions. Therefore, we do not have to include them again. In addition to that, we compared the number of nodes per subprocess and the precision, recall, overshoot and undershoot, which we will explain in more detail.

Precision and *recall* are common measures in information retrieval, used to compare the performance of automated information retrieval to human information retrieval. In our case that is retrieval of subprocesses. In this context precision and recall compare the subprocesses that are found *automatically* to the processes that have been identified by *humans*, where we define a subprocess as a collection of nodes. Precision is commonly defined as the fraction of 'true positives' (i.e. the number of subprocesses that is both found and existing according to humans, divided by the number of subprocesses that is found) and recall is defined as the fraction that represents the 'completeness' (i.e. the number of found existing subprocesses divided by the number of existing subprocesses).

For this context, we consider the way that precision and recall are determined on the basis of exact matches as too strict. Recall that subprocesses are defined by the set of nodes of which they consist. We argue that a subprocess that is automatically retrieved, but from which some relevant nodes are missing as compared to the subprocess determined by human modelers, is not a completely missed match. For example, suppose that we consider that the set of nodes {'receive request', 'fill out request', 'complete request', 'accept request', 'file request'} constitutes a subprocess. If the set of nodes {'receive request', 'fill out request', 'complete request', 'accept request'} is returned automatically, then this is not an exact match, but it does provide useful information to the process analyst that is determining the subprocesses. The process analyst can easily investigate the subprocess and manually complete it; he or she is much better off than when no information was returned at all. Therefore, we define precision and recall in terms of the number of matched nodes that constitute the subprocesses, rather than in terms of the number of (exactly) matched subprocesses.

In addition to the notions of precision and recall, we also introduce *overshoot* and *undershoot* to give a measure of how (im)perfect a subprocess match is. Overshoot is the fraction of found nodes that does no belong in a subprocess; conversely, undershoot is the fraction of nodes that do belong but that were not found. For example, in the example used above the overshoot is 0% (no nodes were found that did not belong) and the undershoot is 20% (the element 'file request' was not found but did belong in the subprocess).

More precisely, we measure precision, recall, overshoot and undershoot as follows. Let N be the set of all nodes in a process (including its subprocesses). Let $\mathcal{P}_{\mathcal{M}} \subseteq \mathbb{P}N$ be the set of all subprocesses that were determined manually by humans and let $\mathcal{P}_{\mathcal{A}} \subseteq \mathbb{P}N$ be the set of all subprocesses that were determined

automatically. Furthermore, let $P_M \in \mathcal{P}_\mathcal{M}$ be a subprocess that was determined manually by humans and let $P_A \in \mathcal{P}_\mathcal{A}$ be a subprocess that was determined automatically. The overlap between P_A and P_M is:

$$\text{Overlap} = \frac{|P_A \cap P_M|}{\max(|P_A|, |P_M|)}$$

We say that P_A is the *most relevant match* for P_M if its overlap with P_M is greater than 0 and there is no other automatically determined subprocess $P'_A \in \mathcal{P}_\mathcal{A}$ with a higher overlap than P_A. Let the function match : $\mathcal{P}_\mathcal{M} \to \mathbb{P}N$ return the most relevant match for each manually determined subprocess, or the empty set if no such match exist.

Precision and recall can then be defined as follows.

$$\text{Precision} = \frac{\Sigma_{P_M \in \mathcal{P}_\mathcal{M}} |P_M \cap \text{match}(P_M)|}{\Sigma_{P_A \in \mathcal{P}_\mathcal{A}} |P_A|}$$

$$\text{Recall} = \frac{\Sigma_{P_M \in \mathcal{P}_\mathcal{M}} |P_M \cap \text{match}(P_M)|}{\Sigma_{P_M \in \mathcal{P}_\mathcal{M}} |P_M|}$$

The F Score is the harmonic mean of the precision and the recall

$$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Overshoot and undershoot are defined as follows.

$$\text{Overshoot} = \frac{\Sigma_{P_M \in \mathcal{P}_\mathcal{M}} |\text{match}(P_M) - P_M|}{\Sigma_{P_A \in \mathcal{P}_\mathcal{A}} |P_A|}$$

$$\text{Undershoot} = \frac{\Sigma_{P_M \in \mathcal{P}_\mathcal{M}} |P_M - \text{match}(P_M)|}{\Sigma_{P_M \in \mathcal{P}_\mathcal{M}} |P_M|}$$

4.5. Evaluation Results

Table 4 and Figure 7 show the results of the evaluation. The figure shows how the subprocess division, using a certain criterion, scores in terms of each of the metrics. The 'combined' criterion is the combination of the 'connectedness' and 'label similarity' criteria as explained in Section 4.3. In the 'original' column, the scores of the original model are repeated to simplify the comparison. If a criterion is parameterized (this holds for the connectedness, label similarity and combined criteria), the results are shown for the parameter-value that leads to the highest F-Score.

The F-Score is the most important metric, because it provides an indication of how well a subprocess division approximates the original manual subprocess division. This metric shows that the connectedness criteria can be used best to approximate a manual division into subprocesses. Interestingly, combining information about connectedness with information about label similarity leads to slightly inferior results, although Figure 7 shows that it leads to a distribution

Table 4: Results of Empirical Evaluation of Subprocess Criteria.

Metric	Block Structuredness	Connectedness	Label Similarity	Combined	Original
Subprocesses	68	49	39	47	20
Nodes per Subprocess (avg)	5.12	4.73	5.95	4.94	8.75
Nodes per Subprocess (min)	1	1	2	1	2
Nodes per Subprocess (max)	103	54	80	57	27
Fan-In	2.95	3.33	6.21	3.83	2.26
Fan-Out	3.32	3.33	6.21	3.83	2.26
FanIn-Out	96.11	122.45	1482.53	215.13	33.42
Depth	5	2	2	2	3
Precision	0.13	0.34	0.22	0.32	–
Recall	0.5	0.37	0.25	0.36	–
F-Score	0.21	0.35	0.24	0.34	–
Overshoot	0.19	0.25	0.68	0.21	–
Undershoot	0.4	0.57	0.72	0.59	–

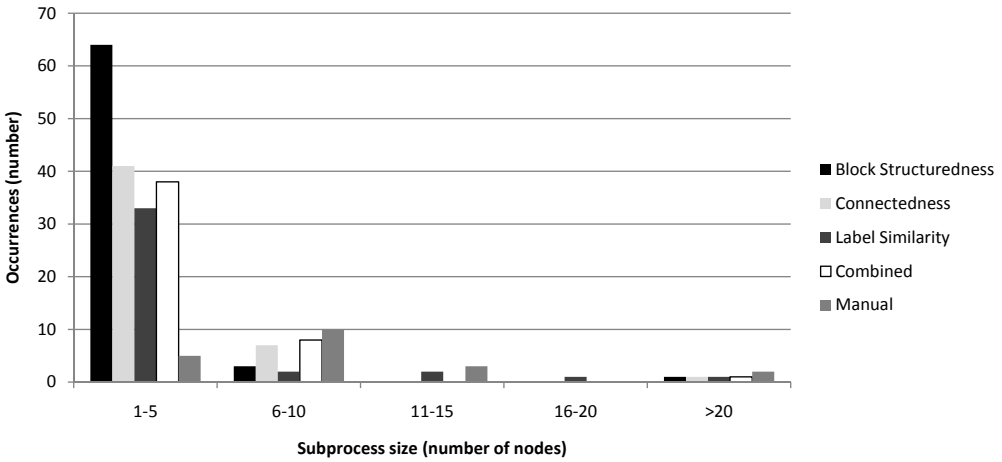

Figure 7: Distribution of Subprocess Sizes

in subprocess size that is significantly better than the one produced by the connectedness criteria.

All of the metrics have high overshoot and undershoot, meaning that for each automatically determined subprocess a large number of nodes is superfluous or missing. For the connectedness criteria, on average 25% of the nodes in an automatically determined subprocess are superfluous and 57% of the nodes are missing. The block structuredness criterion relatively leads to the lowest overshoot and undershoot.

The most striking differences between the automatically and manually derived subprocesses are the number of subprocesses that are created and the sizes of those subprocesses in terms of the number of nodes. The number of automatically determined subprocesses is for all used criteria at least twice that of the number that is created manually. Figure 7 shows the difference between the subprocess sizes' in more detail. For each of the criteria it shows the distribution of the subprocess' sizes, using classes of subprocesses containing between 1 and 5 nodes, 6 and 10 nodes, 11 and 15 nodes, 16 and 20 nodes, and more than 20 nodes. From the table and the figure, we can conclude that under each of the criteria one (very) large subprocess is created and many (very) small subprocesses, while in the manual subdivision mainly mid-sized (6 to 10 nodes) subprocesses exist.

From these results we can point towards (combinations of) criteria that seem attractive to be investigated further for the automated support of dividing a process into subprocesses. Connectedness is the most promising criterion, although it suffers from a high undershoot, which is caused by the fact that it produces a large number of very small subprocesses. Consequently, the results that are produced by this criterion can be improved by merging small subprocesses (i.e. subprocesses containing in the 1 to 5 nodes). Interestingly, label similarity does not present a good criterion for subprocess division; apparently, labels from nodes in the same subprocess are not more similar than labels from nodes in different subprocesses.

These results suggest that there is potential for supporting designers in modularizing their process models. In order to achieve nicely understandable subprocesses the effects of information hiding and additional information costs have to be balanced. Automatic techniques are suitable for proposing candidates of subprocesses for a given model. Yet, the expertise of the designer is required in the end to assess the suitability of a specific modularization.

5. Related Work

The contribution of our work can be related to three streams of research, namely, quality metrics, process model abstraction, and modular workflow execution.

The metrics and criteria that are used in this paper are related to metrics and criteria that are used in software engineering to determine properties of software modules. The dominant metrics that are used in this domain are coupling and

cohesion [22]. Coupling measures the extent to which a module has links with other modules and cohesion measures the extent to which elements within a module are linked. A survey of different classes of those metrics can be found in [62]. Alternative metrics and criteria have been proposed, including the change dependency of module elements [63] and the number and distribution of elements over the modules [64]. Similar to the work described in this paper, work exists that uses clustering techniques to modularize a software system [65]. The work described in this paper is strongly related to the work on modularization in the area of software engineering, and adapts it to the domain of process models. In this way, we also contribute to the area of process model metrics [66]. This research area investigates factors of process model comprehension such as different measures of size and complexity [67, 68, 69, 70]. Ideas of modularity are partially considered in these works, but only for flat process models [71, 72, 73, 38, 74, 75]. Our work extends this research towards a consideration of subprocesses.

Work on process model abstraction relates to our automatic modularization techniques. This work is partially inspired by techniques for matching activities [76], finding similar process models [59], and checking consistency [77]. The algorithm by Polyvyanyy et al. builds on using specific criteria for aggregating activities [78], for instance, if the relative effort of an activity is below a certain threshold. By varying this threshold, the degree of aggregation can be adjusted to the needs of the user. An abstraction approach based on behavioral profiles is presented in [79]. The modeler selects a set of activities that is supposed to be aggregated. Based on the control flow relations of the behavioral profile, this approach generates the control flow of the aggregated model. A different approach based on meronymy relations is presented in [61]. This approach inspects meronymy relations between activity labels to find aggregation candidates. It integrates the problems of finding aggregation candidates and aggregation names. We build on these works to automatically suggest a modularization of the process.

On the execution level, modularity has been shown to be a requirement for distributing the execution of a workflow on different sites. Many of the works in this domain study performance and transactional properties of such a modular and distributed workflow process, e.g. [80, 81, 8]. Recent work on workflows distributed via the internet [82], workflows in a grid environment [83], and workflows in the cloud [84] take a similar perspective. Our research complements these works by demonstrating comprehension benefits of modularity. Yet, it is unlikely that a modular design, which is beneficial from a performance perspective, necessarily leads to better comprehension. It is an open question in how far an easily understandable design relates to good modularity in terms of execution performance.

6. Conclusion

We set out with this paper to address two research problems. The first problem related to the lack of evidence for the usefulness of modularization

through the use of subprocesses in business process models. The results from the controlled experiment that we described in this paper point at the presence of positive effects of subprocesses on the understandability of the model in which they are used. However, the effect may only manifest itself in situations where subprocesses are used on an extensive scale. The most likely explanation that we can present for the additional question why modularization works is that the comprehension of local parts of a process models seems to be improved; subprocesses hide information that is not relevant.

The second problem we addressed related to the lack of theoretically grounded guidelines or dedicated approaches for modularizing a given process model into subprocesses. We compared in an explorative fashion the automated use of three distinctively different types of criteria and one hybrid form for this purpose. A criterion that minimizes the number of edges between subprocesses emerged as the most promising candidate to investigate further.

The results we presented should be considered within the limitations of the experimental and explorative nature of our research approach. Nonetheless, we find it reassuring that positive effects of subprocess usage have been established in an experiment that involved experienced process modelers. In various research, modeler expertise has been established as a critical issue for process modeling projects [85]. Petre observed in her research that expert modelers focus on relevant graphical elements, recognize patterns and disregard irrelevant information [86]. In contrast, novices tend to lack reading and search strategies which result from modeling experience and extensive learning. In that sense, the value of subprocesses is arguably of even greater value in settings where people with low modeling expertise aim to make sense of process models. We aim to follow up on this conjecture in our future research.

Another line of future research relates to the further development of process modularization approaches. At this point, we identified several attractive ingredients for such an approach, in particular with respect to the use of structural characteristics of a process model. Yet, in consideration of the difficulty of finding good modular designs in other domains, it does not seem plausible that a satisfactory, fully automated approach is feasible. Therefore, it seems sensible to get a process modeler "in the loop" in further evaluations of automated approaches, in this way moving the focus from automated modeling towards modeling guidance. This direction would also fit the wider stream of research in the process modeling domain that aims at advanced tools to support process modelers beyond the features that ordinary modeling tools provide, e.g. [87, 88].

References

[1] C. Baldwin, K. Clark, Managing Modularity, Harvard Business Review 75 (5) (1997) 84–93.

[2] R. Langlois, Modularity in Technology and Organization, Journal of Economic Behavior and Organization 49 (1) (2002) 19–37.

[3] R. Eshuis, R. Wieringa, Tool support for verifying uml activity diagrams, IEEE Trans. Software Eng. 30 (7) (2004) 437–447.

[4] W. Aalst, Formalization and Verification of Event-driven Process Chains, Information and Software Technology 41 (10) (1999) 639–650.

[5] R. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in BPMN, Information and Software Technology 50 (12) (2008) 1281–1294.

[6] W. Aalst, A. Hofstede, YAWL: Yet Another Workflow Language, Information Systems 30 (4) (2005) 245–275.

[7] W. Aalst, K. Hee, Workflow Management: Models, Methods, and Systems, MIT press, Cambridge, MA, 2002.

[8] F. Leymann, D. Roller, Workflow-based Applications, IBM Systems Journal 36 (1) (1997) 102–123.

[9] M. Dong, F. Chen, Petri Net-Based Workflow Modelling and Analysis of the Integrated Manufacturing Business Processes, The International Journal of Advanced Manufacturing Technology 26 (9) (2005) 1163–1172.

[10] A. Sharp, P. McDermott, Workflow Modeling: Tools for Process Improvement and Application Development, Artech House, 2001.

[11] A. Lindsay, D. Downs, K. Lunn, Business processesattempts to find a definition, Information and Software Technology 45 (15) (2003) 1015–1019.

[12] M. Ould, Business Processes: Modelling and Analysis for Re-engineering and Improvement, Wiley, 1995.

[13] R. Davis, Business Process Modelling With Aris: A Practical Guide, Springer, 2001.

[14] N. Damij, Business Process Modelling Using Diagrammatic and Tabular Techniques, Business Process Management Journal 13 (1) (2007) 70–90.

[15] B. Weber, M. Reichert, Refactoring process models in large process repositories, Lecture Notes in Computer Science 5074 (2008) 124–139.

[16] H.A. Reijers, J. Mendling, Modularity in process models: Review and effects, in: BPM, Springer, 2008, pp. 20–35.

[17] D. Parnas, On the Criteria for Decomposing Systems into Modules, Communications of the ACM 15 (12) (1972) 1053–1058.

[18] C. Alexander, Notes on the Synthesis of Form, Harvard University Press, 1970.

[19] Y. Wand, R. Weber, On the deep structure of information systems, Information Systems Journal 5 (1995) 203–223.

[20] R. Weber, Ontological Foundations of Information Systems, Coopers & Lybrand and the Accounting Association of Australia and New Zealand, Melbourne, Australia, 1997.

[21] E. Yourdon, L. Constantine, Structured Design, Prentice Hall, 1979.

[22] S. Chidamber, C. Kemerer, A metrics suite for object oriented design., IEEE Transaction on Software Engineering 20 (6) (1994) 476–493.

[23] A. Burton-Jones, P. Meso, How good are these uml diagrams? an empirical test of the wand and weber good decomposition model, in: L. Applegate, R. Galliers, J. DeGross (Eds.), Proceedings of the Twenty-third International Conference on Information Systems (ICIS), 2002, pp. 101–114.

[24] M. Adler, An algebra for data flow diagram process decomposition, IEEE Trans. Software Eng. 14 (2) (1988) 169–183.

[25] A. Basu, R. Blanning, Synthesis and Decomposition of Processes in Organizations, Information Systems Research 14 (4) (2003) 337–355.

[26] B. Weber, S. Rinderle, M. Reichert, Change patterns and change support features in process-aware information systems, in: J. Krogstie, A. L. Opdahl, G. Sindre (Eds.), Advanced Information Systems Engineering, 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007, Proceedings, Vol. 4495 of Lecture Notes in Computer Science, Springer, 2007, pp. 574–588.

[27] T. Basten, W. Aalst, Inheritance of Behavior, Journal of Logic and Algebraic Programming 47 (2) (2001) 47–145.

[28] T. Malone, K. Crowston, J. Lee, B. Pentland, Tools for Inventing Organizations: Toward a Handbook for Organizational Processes, Management Science 45 (3) (1999) 425–443.

[29] J. Desel, J. Esparza, Free Choice Petri Nets, Vol. 40 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, UK, 1995.

[30] W. Sadiq, M. Orlowska, Analyzing Process Models using Graph Reduction Techniques, Information Systems 25 (2) (2000) 117–134.

[31] M. Wynn, H. Verbeek, W. Aalst, A. Hofstede, D. Edmond, Reduction rules for yawl workflow nets with cancellation regions and or-joins, BPMCenter Report BPM-06-24, BPMcenter.org (2006).

[32] S. Jablonski, MOBILE: A Modular Workflow Model and Architecture, in: Proceedings of the International Working Conference on Dynamic Modelling and Information Systems, 1994.

[33] F. Leymann, D. Roller, Production Workflow - Concepts and Techniques, Prentice Hall, 2000.

[34] W. Aalst, Verification of Workflow Nets, in: P. Azéma, G. Balbo (Eds.), Application and Theory of Petri Nets 1997, Vol. 1248 of Lecture Notes in Computer Science, Springer Verlag, 1997, pp. 407–426.

[35] N. Kock Jr, Product Flow, Breadth and Complexity of Business Processes: An Empirical Study of 15 Business Processes in Three Organizations, Business Process Re-engineering & Management Journal 2 (2) (1996) 8–22.

[36] J. Mendling, G. Neumann, W. Aalst, Understanding the occurrence of errors in process models based on metrics, in: R. Meersman, Z. Tari (Eds.), OTM Conference 2007, Proceedings, Part I, Vol. 4803 of Lecture Notes in Computer Science, Springer, 2007, pp. 113–130.

[37] F. Leymann, Workflows Make Objects Really Useful, EMISA Forum 6 (1) (1996) 90–99.

[38] J. Vanhatalo, H. Völzer, F. Leymann, Faster and more focused control-flow analysis for business process models through sese decomposition, in: B. Krämer, K.-J. Lin, P. Narasimhan (Eds.), Service-Oriented Computing - ICSOC 2007, Fifth International Conference, Vienna, Austria, September 17-20, 2007, Proceedings, Vol. 4749 of Lecture Notes in Computer Science, Springer, 2007, pp. 43–55.

[39] J. Mendling, H.A. Reijers, J. Cardoso, What Makes Process Models Understandable?, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), International Conference on Business Process Management (BPM 2007), Vol. 4714 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2007, pp. 48–63.

[40] I. Vanderfeesten, H.A, Reijers, W. van der Aalst, Evaluating workflow process designs using cohesion and coupling metrics, Computers in Industry 59 (5) (2008) 420–437.

[41] T. Green, M. Petre, Usability analysis of visual programming environments: A 'cognitive dimensions' framework, J. Vis. Lang. Comput. 7 (2) (1996) 131–174.

[42] T. Green, Conditional program statements and their comprehensibility to professional programmers, Journal of Occupational Psychology 50 (1977) 93–109.

[43] I. Vessey, Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature*, Decision Sciences 22 (2) (1991) 219–240.

[44] T. A. et al., Business process execution language for web services, version 1.1, Tech. rep., Microsoft, IBM, Siebel Systems, SAP, BEA (2003).

[45] Object Management Group, Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification, dtc/06-02-01, Object Management Group (February 2006).

[46] N. Juristo, A. Moreno, Basics of Software Engineering Experimentation, Kluwer Academic Publishers, 2001.

[47] L. Prechelt, Kontrollierte Experimente in der Softwaretechnik: Potenzial und Methodik, Springer, 2001.

[48] B. Mutschler, B. Weber, M. Reichert, Workflow management versus case handling: Results from a controlled software experiment, in: L. Liebrock (Ed.), Proceedings of the 2008 ACM Symposium on Applied Computing, Volume I, ACM, 2008, pp. 82–89.

[49] J. Cardoso, Poseidon: A Framework to Assist Web Process Design Based on Business Cases, International Journal of Cooperative Information Systems 15 (1) (2006) 23–55.

[50] H. Verbeek, M. van Hattem, H.A, Reijers, W. de Munk, Protos 7.0: Simulation made accessible, in: G. Ciardo, P. Darondeau (Eds.), Proceedings of the 24th International Conference on Application and Theory of Petri Nets, Springer, 2005, pp. 465–474.

[51] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, CRC Press, 2004.

[52] G. Lee, J. Yoon, An Empirical Study on Complexity Metrics of Petri Nets, Microelectronics and reliability 32 (9) (1992) 1215–1221.

[53] J. Mendling, Detection and Prediction of Errors in EPC Business Process Models, Ph.D. thesis, Vienna University of Economics and Business Administration, Vienna, Austria (May 2007).

[54] R. Laue, V. Gruhn, Complexity metrics for business process models, in: W. Abramowicz, H. C. Mayr (Eds.), 9th International Conference on Business Information Systems (BIS 2006), Vol. 85 of Lecture Notes in Informatics, 2006, pp. 1–12.

[55] T. Cook, W. Shadish, D. Campbell, Experimental and Quasi-Experimental Designs for Generalized Causal Inference, Houghton Mifflin, 2002.

[56] J. Vanhatalo, H. Völzer, J. Koehler, The refined process structure tree, in: Proceedings of BPM 2008, Vol. 5240 of Lecture Notes in Computer Science, Springer, 2008, pp. 100–115.

[57] J. Vanhatalo, H. Völzer, J. Koehler, The refined process structure tree, Data & Knowledge Engineering 68 (9) (2009) 793–818.

[58] S. Schaeffer, Graph clustering - survey, Computer Science Review 1 (2007) 27–64.

[59] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, J. Mendling, Similarity of business process models: Metrics and evaluation, Information Systems 36 (2010) 498–516.

[60] B. van Dongen, R. Dijkman, J. Mendling, Measuring similarity between business process models, in: Springer (Ed.), Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE), Vol. 5074 of Lecture Notes in Computer Science, 2008, pp. 450–464.

[61] S. Smirnov, R. Dijkman, J. Mendling, M. Weske, Meronymy-based Aggregation of Activities in Business Process Models, Conceptual Modeling–ER 2010 (2010) 1–14.

[62] J. Eder, G. Kappel, M. Schrefl, Coupling and cohesion in object-oriented systems, Tech. rep., University of Klagenfurt (1994).

[63] G. Gui, P. Scott, Coupling and cohesion measures for evaluation of component reusability, in: Proceedings of the 2006 international workshop on Mining software repositories (MSR '06), ACM, 2006, pp. 18–21.

[64] F. B. e Abreu, M. Goulao, Coupling and cohesion as modularization drivers: are we being over-persuaded?, in: Proceedings of the European Conference on Software Maintenance and Reengineering, 2001, pp. 47–57.

[65] F. B. e Abreu, G. Pereira, P. Sousa, A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems, software maintenance and reengineering, in: Proceedings of the European Conference on Software Maintenance and Reengineering, 2000, pp. 13–22.

[66] J. Mendling, Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, Vol. 6 of Lecture Notes in Business Information Processing, Springer, 2008.

[67] G. Lee, J.-M. Yoon, An empirical study on the complexity metrics of petri nets, Microelectronics and Reliability 32 (3) (1992) 323–329.

[68] M. Nissen, Redesigning reengineering through measurement-driven inference, MIS Quarterly 22 (4) (1998) 509–534.

[69] S. Morasca, Measuring attributes of concurrent software specifications in petri nets, in: METRICS '99: Proceedings of the 6th International Symposium on Software Metrics, IEEE Computer Society, Washington, DC, USA, 1999, pp. 100–110.

[70] J. Cardoso, Process control-flow complexity metric: An empirical validation, in: Proceedings of IEEE International Conference on Services Computing (IEEE SCC 06), Chicago, USA, September 18-22, IEEE Computer Society, 2006, pp. 167–173.

[71] G. Canfora, F. García, M. Piattini, F. Ruiz, C. Visaggio, A family of experiments to validate metrics for software process models., Journal of Systems and Software 77 (2) (2005) 113–129.

[72] E. R. Aguilar, F. García, F. Ruiz, M. Piattini, An exploratory experiment to validate measures for business process models, in: First International Conference on Research Challenges in Information Science (RCIS), 2007.

[73] I. Vanderfeesten, H.A. Reijers, J. Mendling, W. Aalst, J. Cardoso, On a Quest for Good Process Models: The Cross-Connectivity Metric, Lecture Notes in Computer Science 5074 (2008) 480–494.

[74] W. Aalst, K. Lassen, Translating unstructured workflow processes to readable BPEL: Theory and implementation, Information and Software Technology 50 (3) (2008) 131–159.

[75] J. Mendling, H. Verbeek, B. Dongen, W. Aalst, G. Neumann, Detection and Prediction of Errors in EPCs of the SAP Reference Model, Data & Knowledge Engineering 64 (1) (2008) 312–329.

[76] M. Weidlich, R. M. Dijkman, J. Mendling, The icop framework: Identification of correspondences between process models, in: B. Pernici (Ed.), Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings, Vol. 6051 of Lecture Notes in Computer Science, Springer, 2010, pp. 483–498.

[77] M. Weidlich, J. Mendling, M. Weske, Efficient consistency measurement based on behavioural profiles of process models, IEEE Transactions on Software Engineering (TSE)To appear.

[78] A. Polyvyanyy, S. Smirnov, M. Weske, Process model abstraction: A slider approach, in: Proceedings of the 12th International Conference on Enterprise Distributed Object Computing (EDOC), 2008.

[79] S. Smirnov, M. Weidlich, J. Mendling, Business process model abstraction based on behavioral profiles, in: Proceedings of the 8th International Conference on Service Oriented Computing (ICSOC), San Francisco, USA, 2010.

[80] H. Schuster, S. Jablonski, P. Heinl, C. Bussler, A general framework for the execution of heterogenous programs in workflow management systems, in: CoopIS, 1996, pp. 104–113.

[81] J. Puustjärvi, H. Tirri, J. Veijalainen, Reusability and modularity in transactional workflows, Inf. Syst. 22 (2/3) (1997) 101–120.

[82] M. B. Blake, M. N. Huhns, Web-scale workflow: Integrating distributed services, IEEE Internet Computing 12 (1) (2008) 55–59.

[83] O. Ezenwoye, M. B. Blake, G. Dasgupta, S. M. Sadjadi, S. Kalayci, L. L. Fong, Managing faults for distributed workflows over grids, IEEE Internet Computing 14 (2) (2010) 84–88.

[84] K. S. Shams, M. W. Powell, T. Crockett, J. S. Norris, R. Rossi, T. Söderström, Polyphony: A workflow orchestration framework for cloud computing, in: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGrid 2010, 17-20 May 2010, Melbourne, Victoria, Australia, IEEE, 2010, pp. 606–611.

[85] W. Bandara, Factors and measures of business process modelling: model building through a multiple case study, European Journal of Information Systems 14 (2005) 347–360.

[86] M. Petre, Why looking isn't always seeing: readership skills and graphical programming: Cognition and software development, Communications of the ACM 38 (6) (1995) 33–44.

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

[88] A. Koschmider, M. Song, H. A. Reijers, Social software for business process modeling, Journal of Information Technology 25 (3) (2010) 308–322.