Visual Modeling of Instance-Spanning Constraints in Process-Aware Information Systems

Manuel Gall(✉) and Stefanie Rinderle-Ma

Faculty of Computer Science, University of Vienna, Vienna, Austria
{manuel.gall,stefanie.rinderle-ma}@univie.ac.at

Abstract. Instance-Spanning Constraints (ISCs) have raised attention just recently although they are omnipresent in practice to define conditions across multiple instances or processes, e.g., bundling of cargo. It would be crucial to convey ISC information on, e.g., shared instance resources to users. However, no approach for visualizing ISCs has been presented yet. To overcome this gap we analysed literature and derived visualization requirements for constraints on multiple instances of the same or different processes. The proposed language ISC_Viz is based on BPMN-Q and incorporates existing visual notations to reduce the cognitive load on the user. The applicability of ISC_Viz is shown along 114 ISC modeling examples. Moreover, a questionnaire-based study with 42 participants is conducted in order to assess the usability of ISC_Viz.

Keywords: Constraint visualization · Instance-Spanning Constraints · Compliance · Process-aware information systems

1 Introduction

In many cases, business process instances are not executed in an isolated fashion, but share, for example, resources. Restrictions and properties on these interconnections can be expressed based on so called Instance-Spanning Constraints (ISC) [3]. More precisely, ISCs can span multiple instances, but also multiple processes. An example for an ISC spanning multiple instances of the same process is synchronization at a critical resource (e.g., wait until resource is fully loaded). Imposing an order between treatments for a patient in two medical processes is an example for a process-spanning ISC. As shown by a recent study [14], ISC examples can be found for almost any domain. Although ISCs might refer to design time aspects of business processes, the lion's share of ISC examples becomes effective during runtime [3].

Thus a comprehensive ISC support for business processes is indispensable. This includes formalisms to specify ISC and associated verification techniques in order to check for ISC violations. However, as stated in [9] checking constraints by only providing some kind of *violation: yes/no* answer is in general not sufficient. In turn, it is crucial to adequately include users in the constraint checking process as often the users are required to handle constraint violations. In order to be able to deal with constraints and their violations it is necessary that users can understand constraints.

Thus visual modeling languages for constraints in business processes are required. For constraints that do not span any instances or processes, i.e., so called intra-instance constraints (IIC), some proposals for visual modeling languages exist, for example, BPMN-Q [2] and extended Compliance Rule Graphs (eCRG) [7]. These languages, however, were not designed having ISC in mind. Hence, overall, there is no language that supports the visual modeling of constraints spanning multiple instances or processes. However, this would be very important since ISC might be even harder to understand than IIC due to the additional information on the spanning part of the constraints.

Thus this work aims at developing a visual modeling language for ISC. In detail the goals are to

- define requirements for an ISC modeling language.
- elaborate and implement a visual modeling language for ISC.
- show the applicability and usability of the suggested language.

In order to reach these goals, the work at hand follows the design science research methodology (cf. [18]). At first, requirements are derived from existing work on constraints in general and ISC specifically. Existing proposals for visual constraint modeling languages in the business process domain are evaluated along the requirements. As a result an existing language is chosen as fundament for elaborating the visual ISC modeling language ISC_Viz, i.e., the resulting artifacts are a collection of requirements, an assessment of existing languages, and ISC_Viz. ISC_Viz is then evaluated in two ways. Its applicability is shown by modeling the representative ISC for each of the categories introduced in [3] and modeling the complete ISC example data set of 114 real-world ISC presented in [14]. The usability of the language is evaluated based on a user study. Stakeholders of the proposed solution can be process and constraint designers, auditors, as well as process participants.

The paper is structured as follows. Section 2 derives requirements for a visual ISC modeling language and Sect. 3 evaluates existing constraint modeling languages along these requirements. In Sect. 4, a visual modeling language for ISC is proposed. Section 5 presents the evaluation. In Sect. 6 related approaches are discussed and Sect. 7 closes with a summary.

2 Requirements

To create a visual language that fits the needs of ISC we derive requirements from existing work. *Ly et al.* [9] introduces a framework for Compliance Monitoring Functionalities (CMF). This framework consists of three groups of requirements, i.e., modeling requirements, execution requirements and user requirements. In the following, we focus on CMF modeling requirements as input for deriving requirements on modeling instance spanning constraints (ISC). The modeling requirements consist of three functionalities referring to time, data, and resources. These three functionalities can be mapped to requirements for modeling ISC.

1. *Time* enables the specification of temporal constraints, e.g., for a specific moment in time and period of time.
2. *Data* can be restricted to one instance or shared between multiple instances of different processes. We differentiate between two data types. *Process data* consists of input and output data which is read or written when a task is executed. *Execution data* is created by executing instances and can be seen as meta data, i.e., how many instances of a certain process are currently running.
3. *Resources* can be restricted i.e. one resource can be accessed by a maximum of five instances simultaneously.

These requirements are refined within the IUPC [11] framework and the CRISP project [3]. The IUPC framework helps to identify process constraints based on several criteria. These criteria are used to define our ISC requirements.

4. *Behavior:* describes in which order tasks are executed. A compliance rule engine might, for example, enforce that a certain task has to be executed before another.
5. *Structural Pattern:* defines the connection between constraint and process. A structural pattern consists of one or multiple tasks.
6. *Trigger:* defines when the constraint is checked. A constraint can be checked based on time and structural pattern. Time can be a specific point in time or a recurring check every day at a certain time. Structural pattern is triggered before or after a task is executed and might involve data and or resources.
7. *Interoperability:* describes that one constraint might span multiple of these requirements, e.g., a booking process has to be executed within a certain time and depends on a specific resource. The proposed visualization shall be able to comprehend different constraint types within one visualization without additional semantics.
8. *Spanning Processes:* ISC can span single or multiple processes [3]. A constraint only referring to one process has to span multiple instances in order to be considered an ISC.
9. *Spanning Instances:* Typically, ISC impose constraints on multiple instances. A constraint that refers to instances in a separate way, i.e., does not span any instances, is referred to as intra-instance constraint (IIC). Taking a design time perspective, ISC can also span multiple processes, but no instances.
10. *Action* refers to the behavioral part of the IUPC framework e.g. synchronization between process instances. Such a synchronization needs two actions *wait* to stop all involved instances before or after a certain activity and *start* to start execution of the synchronous activities.

These 10 requirements build the foundation for evaluating and selecting a visual modeling language. The selected language is then extended to support ISC.

3 Analyzing Visual Constraint Modeling Languages

Our goal is to propose a visual modeling language for ISC that can be used for IIC. For this reason we take a look at current visual constraint modeling languages in the area of business processes and evaluate them along the ISC requirements set out in Sect. 2. The following sections contain a brief description of each language and show a visual model of an IIC for illustrative purposes. As representative IIC we are using an example from a study on constraint visualization, i.e., *"c5: The testing has to be followed by an approval and the integration. Additionally, no changes shall take place between the approval and the integration."* [10]. Assume that this IIC is enacted on the BPMN model depicted in Fig. 1.

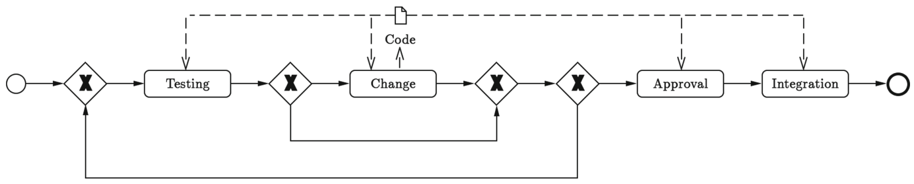

Fig. 1. Code testing example c5 from [10] modeled with BPMN.

To the best of our knowledge, BPMN-Q [2] and eCRG [7] are the only visual modeling languages for constraints in the business process context. Hence both languages are selected as candidates for extension towards modeling ISC and evaluated along the harvested requirements in the following.

3.1 BPMN-Q

BPMN-Q [2] extends BPMN to enable visual query modeling based on a set of processes. However, BPMN-Q can be easily adjusted for compliance checking and hence constitutes a candidate language for visual ISC modeling. One of the strengths of BPMN-Q is that it does not introduce a completely new visual notation as it is based on BPMN. There are a few additional language elements to be learned. In the initial version of BPMN-Q *Awad* focus on control flow. In [2] the approach is extended towards visual modeling of data and resources [1]. Currently the notion of time constraints is not integrated with BPMN-Q. However BPMN allows for modeling time-related information such as point in time and time spans. Overall, this covers the modeling requirements for ISC modeling. Finally, BPMN-Q can be mapped onto past linear time logic (PLTL).

Figure 2 visualizes the constraint *c5* with BPMN-Q syntax.

3.2 Extended Compliance Rules Graphs – eCRG

Compliance Rule Graphs (CRG) [10] initially focused on visually modeling control flow related constraints. The approach was extended (eCRG) [8] to enable

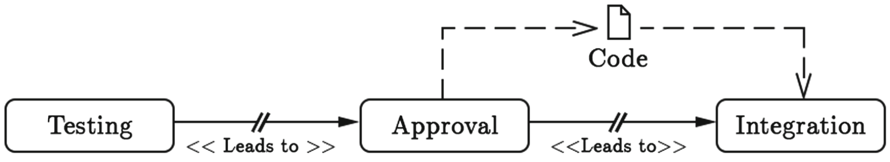

Fig. 2. Code testing example *c5* modeled with BPMN-Q

modeling of time, data, and resource constraints. Time constraints can be modeled in eCRGs in different ways, e.g., by modeling a particular point in time or so called time distance. The latter allows for modeling time constraints for single and multiple tasks. eCRG does not enable the modeling of *execution data*. Figure 3 visualizes constraint *c5* in eCRG syntax.

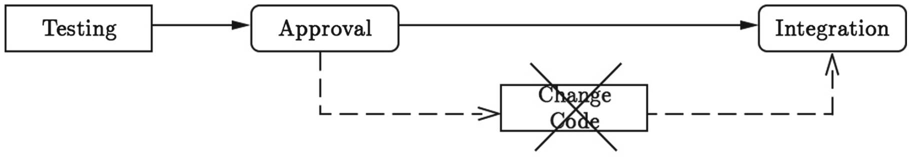

Fig. 3. Code testing example *c5* modeled with E-CRG

3.3 Requirements Analysis

As shown in Table 1 BPMN-Q and eCRG fulfill some of the requirements. Both deal with time and data constraints in a different way, but do not allow for modeling the full capabilities that are required for ISC such as execution data. None of the languages enables the modeling of "spanning information" at process and instance level. Moreover, it is not possible to model that certain actions are to be enforced.

By using the existing BPMN visualization for time the shortcoming of BPMN-Q compared to eCRG is minimal. An advantage of BPMN-Q might be that the underlying process modeling notation BPMN is known to a broader audience. As information on the underlying process models plays a vital role for ISC (even more than for IIC), we finally opted for BPMN-Q as the fundament for developing ISC_Viz. In particular, this requires to propose an extension covering time/data/trigger/action visualization and the instance spanning part of constraints.

4 ISC_Viz

The requirements analysis revealed that BPMN-Q needs extensions with respect to *Trigger*, *Spanning Processes*, *Spanning Instances*, and *Action* (cf. Table 1)

Table 1. The first column references each requirement defined in Sect. 2. For each requirement a "X" marks that this requirement is satisfied, a ∼ marks partially satisfied requirements.

Requirements	BPMN-Q	E-CRG
Time	∼	X
Data	∼	∼
Resources	∼	∼
Behavior	X	X
Structural pattern	X	X
Trigger	∼	∼
Interoperability	X	X
Spanning processes		
Spanning instances		
Action		

in order to express ISC-related information. For *Trigger* visualization existing BPMN symbols can be used. For *Spanning Processes and Spanning Instances* and *Action* additional visualization concepts must be proposed. Specifically, this means to enrich a graph with additional information. In order to not reinvent the wheel, experience from visualization approaches in the business process domain are considered, i.e., the work on visualizing differences between business process in [4]. In this work, nine visualization possibilities, e.g., shapes, color, and symbols, were analyzed in order to suggest a generic visualization that can be applied to a wide range of process model types.

4.1 Visualizing Spanning Processes and Spanning Instances

As the goal is to extend BPMN-Q with information on *Spanning Processes* and *Spanning Instances* we need a visual style that can be incorporated into the language. For example, adding new shapes for process and instance spanning information does not seem to be sufficient due do the number of new shapes that is necessary for expressing, for example, spanning data elements, time elements, and task dependencies.

We [4] emphasize that colors and symbols are suitable to visualize differences between multiple processes. Color is a visual element that is currently not used within BPMN-Q. By using color to express information on *Spanning Processes* and *Spanning Instances* two additional visual elements are introduced. The "standard" black version for IIC remains the same and two new versions using different colors are introduced for ISC, i.e., *Green* visualizes *Spanning Instances*, while *Blue* visualizes *Spanning Processes.*

4.2 Visualizing Actions

Besides color *we* [4] recommend the usage of symbols to show differences between graphs. For visualizing actions such symbols offer various advantages over color.

An advantage is that different types of actions can be expressed as various symbols are available. Colors are technically not limited, but the cognitive perception is restricted with respect to distinguishing colors. Another advantage of using symbols is extensibility. So far a set of actions for one subject has been described such as start and wait actions connected with an activity. However, further actions are conceivable. In this case these new actions can be visualized using additional symbols. The set of symbols suggested in this paper is known from user interfaces like execution engines and media player controls. Figure 4 depicts the symbols suggested in this work for the action part described in current literature [11,15].

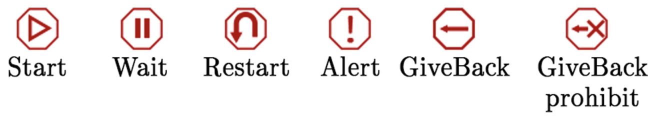

Fig. 4. Selected actions and associated symbols (Color figure online)

4.3 Visualizing Trigger

For trigger visualization symbols from BPMN are used. For constraints involving data and resources we use *conditional.Conditional trigger* describes the integration of external business rules which is suitable for compliance rules. Data and resources checked within the trigger are visualized with arrows leading to the trigger. Time constraints are visualized with the BPMN *Timer*. The *timer trigger* allows for expressing points in time, time spans, and timeouts. In order to differentiate between intra-instance and instance-spanning we use a different color for ISC triggers.

4.4 Complete Visualization

Overall, the proposal is to visualize the spanning part of a constraint with color, the more complex actions with symbols, and the trigger with BPMN symbols.

Illustrating the extensions to BPMN-Q Fig. 5 shows a process model and the ISC *"Wait until centrifuge is filled."* [14]. At first, some explanation is given on the meaning of the colors. Then the general structure of the language is described.

- Green activity shows that this constraint spans multiple instances.
- Purple shows a trigger with a data constraint "execution data". Centrifugation is only done when the centrifuge is full. Information if the centrifuge is full can be derived from other instances. When there are 5 mixtures waiting for centrifugation (execution data) and the centrifuge allows for 6 mixtures then all 6 instances resume work after the 6th mixture is put into the centrifuge.

– Red are the actions as they perform critical tasks and influence the process execution. In this example all instances are stopped before centrifugation until the centrifuge is full. When the centrifuge is full all instances resume their work.

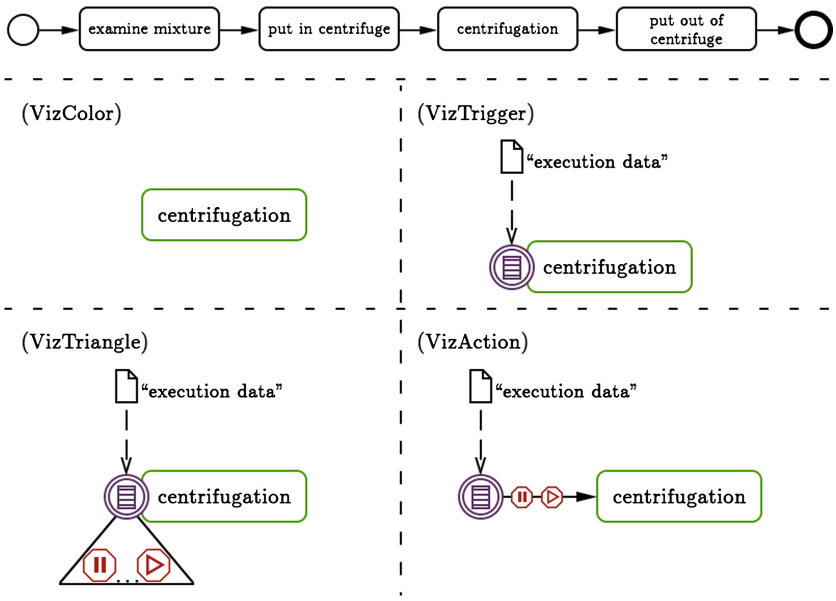

Fig. 5. Top: shows a process model for a centrifugation process. Bottom: visualizes an ISC with trigger and actions. (Color figure online)

As can be seen from Fig. 5 the constraint is modeled in four ways as it is not yet answered how many details shall be shown to the user. These four types are referenced by the following evaluation as follows.

– *VizColor*: visualizes the spanning part of the constraint.
– *VizTrigger*: is based on *VizColor* and adds the *trigger* visualization.
– *VizTriangle*: is based on *VizTrigger* with additional *action* visualization. To show the actions within a triangle is a way to keep the constraint vertically small when multiple activities with diverse trigger and actions are involved.
– *VizAction*: describes the same information as *VizTriangle*, but with a different visualization of the *action* part. Here, the actions are modeled and visualized in a more process-oriented way and thus connected by edges.

5 Evaluation

We evaluate the ISC_Viz proposal in two ways. First a questionnaire is addressing visual detail and understanding of modeled constraints. Secondly, the applicability is illustrated by modeling examples for each category introduced by *Fdhila et al.* [3].

5.1 Questionnaire

The questionnaire was designed aiming at two goals. The first goal was to identify which of the proposed visualizations is preferred by the participants. Second goal was to identify how certain BPMN-Q language extensions such as actions and trigger are understood by the participants.

Method. The initial draft of the questionnaire was designed based on the guidelines by *Porst* [13]. Structured with an introduction, questions targeting the visualization and gathering empirical data. The introduction briefly explained ISC and provided an example process consisting of three constraints where each one was visualized with all four types presented in Sect. 4.4. First question of the questionnaire was about visual preference. Based on three example constraints participants had to mark their preferred type. Further questions from this section refer to, for example, assignment of attributes (spanning, trigger, actions) for a given constraint, based on a visualization selecting an appropriate textual description, ranking of visualizations, animation, and color validation. The questionnaire concludes with demographic questions about age, employment, and gender.

This draft was refined with a 2-stage pretest. In the first stage the questions were discussed with a group of peers familiar with ISC. Goal of this stage was to validate the understanding and goal behind each question. Outcome of this stage was a refinement of question and answer wording. The second stage consisted of a test where three participants from the target audience performed the questionnaire. These three participants were allowed and encouraged to ask questions, but those questions were just noted and not answered. Based on this stage the introduction was changed as participants were confused what the difference between trigger and action is.

As ISC visualization is a new and arising research topic there does not exist a large group of experts for participating in the questionnaire. Therefore the target audience was set to participants familiar with process modeling visualizations as this is one of the foundations our visualization builds upon. On a scale from 1 (expert) to 7 (never worked with process models) the participants rated themselves with a mean (2.90) and median (3). In total 42 computer science students from three master courses participated in the questionnaire. The majority of participants (59.52%) were men, (30.95%) women, (7.14%) with no answer and (2.39%) other. The questionnaire and respective answers are available from[1].

Results. The questionnaire was printed and while all questions besides the demographic questions are mandatory it is up to the participant if the question is answered. From the set of 42 participants two missed to answer a question. The answers of these two participants were retained within the dataset. For affected evaluations we will outline that answers are missing.

[1] http://gruppe.wst.univie.ac.at/projects/crisp/index.php?t=visualization.

The goal is to measure how well the ISC_Viz proposal is understood with focus on the extensions, i.e., *Spanning Processes*, *Spanning Instances*, *Trigger*, and *Action*. In order to evaluate the understanding the participants had to categorise six visual ISC examples. For each example the participants had to check a range of constraint properties, for example, *Spanning* or *Not Spanning*. These properties are reflected on the X-Axis of Fig. 6. On the Y-Axis the accumulated mean number from the 6 example processes and according percentages are shown. High values on the Y-Axis show a high accordance with our proposed ISC_Viz language. When a visualization shows a property and a participant marked it then this counts as one participant. When a visualization shows a property and a participant has not marked this property then this does not count. Contrary to this when no property is shown and a participant marked one this does not count and when the participant did not mark the property it counts. For example, Fig. 5 *VizAction* is given we expect a participant to mark the following properties *Spanning*, *Data Constraint*, *Action Wait* and *Trigger Before*. This leads us to a result where we can see that symbols for action (*Action_Give_back*, *Action Restart*, *Action Wait*) are in accordance by a mean of 35.5 (84.52%) participants. *Trigger After* shows a mean of 33.83 (80.56%) participants in accordance and seems to be understood very well while *Trigger Before* seemed to be confusing with 28.83 (68.65%) accordance. *Time Constraints* themselves seem to be clear 38.33 (91.27%) to the participants while data constraints show an accordance of 25 (59.52%). This is a surprising result as the data constraint is not changed in visual representation compared to BPMN.

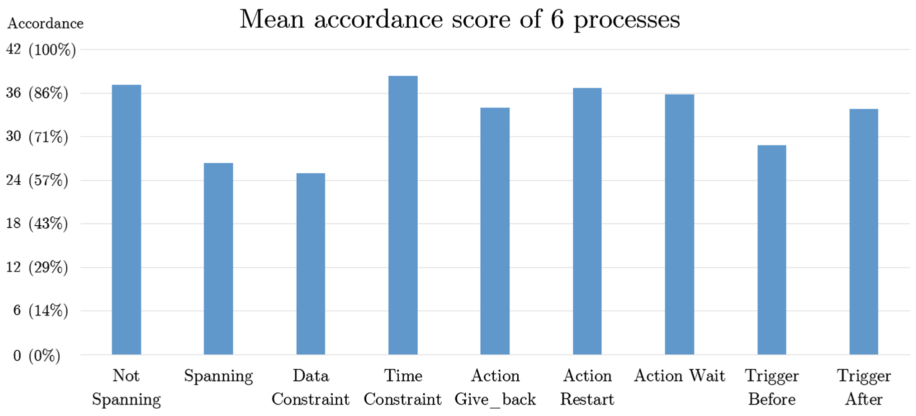

Fig. 6. Results of questionnaire: understanding of ISC_Viz.

As their first task the participants had to select their preferred type based on a short textual description of a constraint. The descriptions increased in difficulty. The first example was a simple constraint spanning instances, the second and third example constraint span processes. First and second descriptions used the same type of data element (execution data) while the third used a specific data element. Table 2 shows the distribution over 41 participants. Each participant was allowed to select one type per description. *VizColor* was chosen in 6.5% of the cases. *VizTriangle* sums up to 23.58% and is evenly distributed among the examples. *VizTrigger* sums up to 33.33% and favours the simple examples while *VizAction* clearly favours the complex example3 with a total of 36.59%.

Table 2. For each example column the distribution is shown across all 4 types.

Type	Example 1	Example 2	Example 3	Sum	Percentage
VizColor	5	2	1	8	6.50%
VizTrigger	16	18	7	41	33.33%
VizTriangle	10	8	11	29	23.58%
VizAction	10	13	22	45	36.59%

While the classification above was shown at the beginning of the questionnaire the following ranking was conducted after working through various ISC examples. The task was to rank the types beginning from 1 (best) to 4. Table 3 summarizes these results. Analysis with the Friedman test show an order preference with $\chi^2(3) = 59.69, p < .0001$, with *VizAction* being best ranked mean (1.64) followed by *VizTrigger* (1.95), *VizTriangle* (2.79) and concluding with *VizColor* (3.62). However an analysis with Wilcoxon signed-rank test shows no significant difference between *VizAction* and *VizTrigger* ($Z = -1.27, p = .102$). When comparing the second and third example based on Friedman test with Wilcoxon test a significant difference is observed ($Z = -3.02, p = .0.00126$).

Further into the questionnaire the participants had to answer if expand functions are preferred. Extending *VizTrigger* to *VizTriangle* is preferred by 47.61% and *VizTrigger* to *VizAction* is preferred by 66.67%.

Table 3. Each column represents the ranking achieved by the type. Low ranking represents preferred visualization.

Rank	*VizColor*	*VizTrigger*	*VizTriangle*	*VizAction*
1	1	15	4	22
2	4	14	11	13
3	5	13	17	7
4	32	0	10	0
Sum	152	82	117	69
Mean	3.62	1.95	2.79	1.64

The visualization part of the questionnaire concluded with a question what a green activity expresses. From 42 participants 35 (83.33%) answered instance spanning, 6 (14.29%) marked process spanning, and 1 (2.38%) did not answer the question. This shows that the participants understood that the spanning part of the constraint is expressed by color.

In summary these results suggest that either *VizAction* or *VizTrigger* are suited for visualizing constraints. Based on the fact that *VizAction* ranks better in both cases, the first more intuitive rating and the second ranking we suggest the usage of *VizAction* for visual modeling of ISC.

For creating a modeling tool for ISC_VIZ we suggest the usage of expand and collapse interaction between *VizTrigger* and *VizAction*.

As another result, the name for the *trigger before* and *trigger after* are changed to *conditional before* and *conditional after* to be more precise and to have a clear distinction to the *timer* trigger.

Table 4. For each category by *Fdhila et al.* [3] we picked one example from *Rinderle-Ma et al.* [14] which will be modeled with ISC_Viz

Context	Requirements	Rule
Multiple	Multiple	"A user is not allowed to execute more than 100 tasks (of any workflow) in a day"
Multiple	Single	"Maximal KWP-2000 Connections The number of connections to KWP2000 should not exceed 10"
Single	Multiple	"There should not exist more than one instance of W such that the input parameters (say loan customer) is the same and the loan amount sums up to $100K during a period of one month"
Single	Single	"Wait until centrifuge is filled"

5.2 Example Based Evaluation

Based on the results from the questionnaire the following examples are visualized with *VizAction*. Our examples are picked from a meta study on run-time ISC [14]. Each example fits one category of the classification introduced by [3]. The classification is divided into a category for design-time and four categories for run-time. Figure 7 shows how these categories differentiate from each other. Context expresses the spanning part of constraints. A constraint is considered *single* spanning when the constraint spans only processes or instances and *multi* spanning when it spans both. Modeling requirements are for example time, data and resources. Modeling requirements are considered *single* when a constraint uses none or one modeling requirement. Constraints that involve more then one modeling requirement are expressed as *multi*. For our evaluation we pick one example from the meta study per category and model these constraints with our ISC_Viz language. To give a comprehensive view all constraints from Table 4 are visualized within Fig. 7.

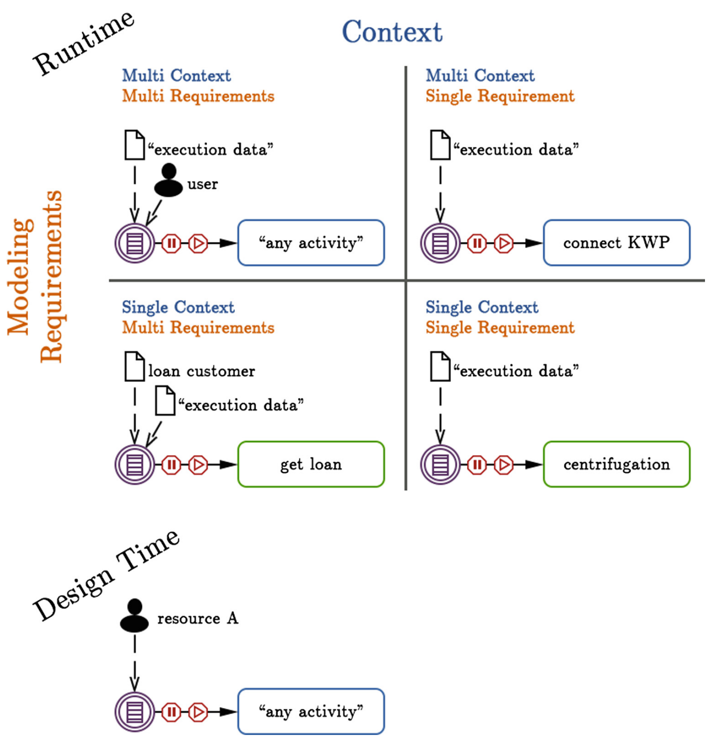

Fig. 7. Each of the categories from [3] is represented with one example plus one design time example.

As *we* [14] do not provide any design time ISC, in addition, the following example constraint is introduced:

A resource is shared among processes but cannot be accessed at the same time by multiple instances.

For a better understanding of possible violations of the design time ISC, assume the following two situations reflected in the associated process models. *Process one uses the resource every day at 8 pm for 30 min. Process two needs to use the resource every Monday at 8 pm for two hours.* With these two textual descriptions it is clear to see that a violation will happen every Monday when two processes try to access the resource at the same time.

Figure 7 depicts the ISC_Viz models for all five ISC examples. It can be seen that ISC_Viz enables the modeling of ISC representatives for all categories. In addition to the examples provided in Table 4 all 114 ISC from the meta study [14] are modeled with ISC_Viz. The models can be found here[2].

[2] http://gruppe.wst.univie.ac.at/projects/crisp/index.php?t=visualization.

6 Related Work

Dealing with business process compliance is a broad research field that can be divided into design time [5] and runtime [9]. Both categories consider various perspectives [16], i.e., control flow, time, data, and resources. For a comprehensive view the iUPC Framework [11] was developed. Process-Aware Information Systems are executing multiple instances of various processes simultaneously. This simultaneous execution allows for further development of business process compliance from IIC towards ISC [6,12,17]. Across several domains ISC examples [14] are collected. Event Calculus is proposed and evaluated for formalizing ISC [3]. These approaches are fundamental to create a visual ISC modeling language. Intra instance constraints are visually modeled in several ways [2,10]. These approaches consider various perspectives, i.e., time, resource and data. However they focus on intra-instance constraints and do not allow for visually modeling ISC.

7 Conclusion and Outlook

This paper introduces the visual modeling language ISC_Viz for IIC and ISC. Specifically the latter has not been addressed in the literature. ISC_Viz is based on BPMN-Q for the intra-instance part and extended by two visual styles, i.e., colors and symbols. These styles enable to model spanning information (i.e., between instances and processes) as well as action and trigger information. ISC_Viz was evaluated with respect to its applicability and usability. More precisely, 114 real-world ISC examples were modeled using ISC_Viz. Moreover, the approach was evaluated with 42 participants. The latter showed that ISC_Viz can be understood with little training. This proposal builds the basis for future research on visualization of ISC, for example, the visualization of compliance violations.

Acknowledgment. This work has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-072.

References

1. Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for data aware compliance rules. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC/ServiceWave -2009. LNCS, vol. 5900, pp. 500–515. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10383-4_37
2. Awad, A.: BPMN-Q: a language to query business processes. In: Proceedings of EMISA 2007, pp. 115–128 (2007)
3. Fdhila, W., Gall, M., Rinderle-Ma, S., Mangler, J., Indiono, C.: Classification and formalization of instance-spanning constraints in process-driven applications. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 348–364. Springer, Cham (2016). doi:10.1007/978-3-319-45348-4_20

4. Gall, M., Wallner, G., Kriglstein, S., Rinderle-Ma, S.: A study of different visualizations for visualizing differences in process models. In: Jeusfeld, M.A., Karlapalem, K. (eds.) ER 2015. LNCS, vol. 9382, pp. 99–108. Springer, Cham (2015). doi:10.1007/978-3-319-25747-1_10
5. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74974-5_14
6. Heinlein, C.: Workflow and process synchronization with interaction expressions and graphs. In: International Conference on Data Engineering, pp. 243–252 (2001)
7. Knuplesch, D., Reichert, M., Kumar, A.: Visually monitoring multiple perspectives of business process compliance. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 263–279. Springer, Cham (2015). doi:10.1007/978-3-319-23063-4_19
8. Knuplesch, D., Reichert, M., Ly, L.T., Kumar, A., Rinderle-Ma, S.: On the formal semantics of the extended compliance rule graph. Technical report UIB-2013 - 05, Ulm University, Ulm, April 2013
9. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: Compliance monitoring in business processes: functionalities, application, and tool-support. Inf. Syst. **54**, 209–234 (2015)
10. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 9–23. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13094-6_3
11. Mangler, J., Rinderle-Ma, S.: IUPC: identification and unification of process constraints. CoRR abs/1104.3609 (2011). http://arxiv.org/abs/1104.3609
12. Pflug, J., Rinderle-Ma, S.: Dynamic instance queuing in process-aware information systems. In: Symposium on Applied Computing, pp. 1426–1433 (2013)
13. Porst, R.: Fragebogen. Ein Arbeitsbuch. VS Verlag für Sozialwissenschaften, Wiesbaden (2008)
14. Rinderle-Ma, S., Gall, M., Fdhila, W., Mangler, J., Indiono, C.: Collecting examples for instance-spanning constraints. Technical report abs/1603.01523, CoRR (2016)
15. Rinderle-Ma, S., Mangler, J.: Integration of process constraints from heterogeneous sources in process-aware information systems. In: International Workshop on Enterprise Modelling and Information Systems Architectures, pp. 51–64 (2011)
16. Sadiq, S., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75183-0_12
17. Warner, J., Atluri, V.: Inter-instance authorization constraints for secure workflow management. In: Symposium on Access Control Models and Technologies, pp. 190–199 (2006)
18. Wieringa, R.: Design Science Methodology for Information Systems and Software Engineering. Springer, Heidelberg (2015)