1 Impacts of Change Operation Delete

Delete Context Removing a context from an ISC will prevent it from monitoring future events of the corresponding process execution. This does not block such events from being handled as they still might be used by different ISC. Such a change implies a modification of the ISC structure (static impact) to enable filtering the appropriate event types, i.e., the router. Using the router on top of the inference engine provides a more generic solution that is not dependent of the rule matching algorithm, e.g., Rete [1].
Versioning: the old version is not removed until all past instances terminate. The new version is added in parallel, and all its shared variables are reinitialized. In the example of Fig. 1, the old counter for both melanoma and dermatitis patients is kept for the old version, and a new counter for dermatitis only is created for the new version. The latter should not include dermatitis patients that arrived before the change time t_c. The router is responsible for correlating facts with the corresponding version.
Migration: old shared variables are reused as they include facts of past process instances (started before t_c). This entails counting dermatitis patients arrived before and after t_c for the day. Migration prevents querying the working memory (a base that contains all facts) from the scratch and reuses matching results of the ISC_old. Indeed, the ISC instance does not only store information about the patients counter but also refers to all facts that are consumed by it (references to melanoma and dermatitis patients). This simplifies the recounting during migration.
Clean state: will utilize the new version only, remove the old one and all its variables. All facts of old process instances are ignored. A routing is necessary to ignore future events related to old process instances.
Context change might result in inconsistencies with respect to behavior/actions that already fired before t_c. Indeed, actions might block a set of process instances when a constraint matches (e.g., synchronization). When changing the corresponding ISC, compensation actions need to be enacted to unblock those process instances. This problem needs to be dealt with for all strategies, but most particularly for clean state as it ignores past process instances. For example, assume that the deletion of the context melanoma treatment process from the ISC (cf. Fig. 1) takes effect at 12pm and the limit of 60 patients (e.g., 40 for blood test and 20 for urine test) was already reached by that time. The change

DELETE	BEFORE	AFTER
CONTEXT	The number of patients accepted for dermatitis and melanoma treatments should not exceed 60 per day.	The number of patients accepted for dermatitis treatments should not exceed 60 per day.
CONNECTION	At 12:00 and 14:00, the average readout of all meters should have a value less than X	At 12:00, the average readout of all meters should have a value less than X
CONDITION	When starting the read-out at 00:00, 99% of all meters should be read out within 6 hours and readout value should not exceed X.	When starting the read-out of 00:00 values, 99% meters should be read out within 6 hours.
BEHAVIOR	For 100 (simultaneous) ad hoc readouts , if 10 meter checks exceed 6 hours then send an alert and stop the readouts.	For 100 (simultaneous) ad hoc readouts , if 10 meter checks exceed 6 hours then send an alert.
ISC	When starting the read-out of 00:00 values 99% of all meters should be read out within 6 hours.	-
ADD	**AFTER**	**BEFORE**

Figure 1: Change Operations: Delete and Add Examples

will reduce the total number of patients to 40, and consequently the status of the ISC will change from fired to not fired, and it will be possible to accept new blood test patients. The problem, is that patients that were rejected before the change and after the ISC has fired, need also to be considered through the actions if possible. Note that compensation actions are domain and scenario specific and it is not always possible to identify compensation actions.

Delete Connection Deleting a connection reduces the number of event types to be monitored by the changed ISC. The event type might refer to a process task, resource attribution or a time trigger. If there exist a context of the changed ISC that depends solely of that connection, then the latter will also be deleted as a consequence. Deleting connections related to process events follows the same procedure as delete context. However, deleting a time trigger might have less impact on the ISC structure, and consequently migration or versioning become simpler. For example, in Fig. 1, the average of meter readouts changed from being checked at 12:00 and 14:00 to only 12:00. This means that the ISC is triggered only one time instead of two, an consequently readouts coming after 12:00 will not be considered. If the change becomes effective after 12:00 then the shared variables and the ISC instance will be reinitialized (the read out counter is set to zero) for the day, unless they are used by another ISC.

Delete Condition An ISC might involve multiple conditions. Deleting a condition releases a constraint on the events/facts. This impacts directly the structure of the ISC, which should be dissociated from that condition. The condition structure is solely removed if it is not used by any other ISC. While versioning will keep using that condition for future events of process instances started before t_c, migration will reuse facts/events that already checked by the old ISC version. If we consider $c_1 \wedge c_2$ as the conditions of the changed ISC and c_1 is the deleted one, then the set of facts that already matched $c_1 \wedge c_2$ will still match c_2 alone. Thus, migration can reuse the current state of the old version and all matched facts. However, it is possible to have facts that were not considered before the change because they did not match c_1, which is no longer required. Consequently it becomes necessary to also check whether those facts are still in the working memory and reuse them. Note that the ordering of conditions is also important in migration. For example, if c_2 is the deleted condition, then there is no need for re-querying the working memory as we already keep track of all facts that matched c_1 before they are filtered by c_2. In Fig. 1, the ISC instance initially fires if the total readout value exceeds X or 99% of all readouts did not finish within 6 hours. Then, consider that the value X has been reached before the 6 hours has elapsed, and the change operation (i.e., deletion of the threshold x condition) becomes effective after the ISC has fired and before the 6 hours constraint expires. In this case it is possible to reconsider the ISC status and continue checking the facts in order to meet the second condition; i.e., 99% of readouts should have been finished within 6 hours. Firing an ISC entails the execution of the actions specified in its behavior part. As such, migrating a fired ISC after a deletion of a condition not only reconsiders the facts associated with it and those in the working memory, but also the actions that are already executed as a consequence of the firing. This goes from

a simple alert that cannot be undone to actions on process tasks that can be migrated depending on the process scenario. For example an action of type *wait* on a task can be undone by a action of type *continue*.

Delete Behavior As aforementioned, a behavior is an action that is executed if an ISC has fired; e.g., stop or wait a task. Deleting an action does not have any effect on the rule structure in the ISC monitor as it does not alter its condition or linkage, but might have effects on the process instances affected by the action. For example, removing the action stop the readouts will have impacts on the process instances that have been stopped as a consequence of this action. In this case, the action can be overridden by continuing the stopped readouts. However, not all actions can be undone or revoked.

Delete ISC The deletion of an ISC releases restrictions on the process instances. This entails that three parts constituting the ISC will be removed. The versioning strategy will keep the ISC until all old instances terminate, while migration will behave as clean state and remove restrictions on both old and new process instances.

References

[1] Forgy, C.: Rete: A fast algorithm for the many patterns/many objects match problem. Artif. Intell. 19(1), 17–37 (1982)