Business Process Dependencies Formal Definitions

Gal Engelberg[1,2][0000−0001−9021−9740], Moshe Hadad[1,2][0000−0002−9315−6260], and Pnina Soffer[1][0000−0003−4659−883X]

[1] University of Haifa, Abba Khoushy Ave 199, Haifa, 3498838
spnina@is.haifa.ac.il
[2] Accenture Labs, Tel Aviv, Israel
{moshe.hadad,gal.engelberg}@accenture.com

1 Running Example

The business process model depicted in Figure 1 is a credit request evaluation process, introduced by [2]. The process model outlines the steps involved in assessing, verifying, and approving or rejecting credit applications. The process initiates when a credit request is received. The Verify task examines the requester's information and the requested amount. Based on the verification result and amount threshold (above or below 5000), the request proceeds to either an Advanced Assessment for high amounts or a Simple Assessment for lower amounts. Each assessment yields a decision, which, if unfavorable, prompts a Renegotiate Request task; otherwise, it moves to the final decision stage.

Data elements such as verification status, amount, interest, requester's first name initial, and final decision are used to guide the process flow. Decision points (gateways) are utilized to ensure the workflow conforms to specified business rules, including notification handling if the decision is negative.

The workflow leverages specific resources, with roles defined as Assistants and Experts. Assistants (Pete, Mike, Ellen) handle initial verification and customer notifications, while Experts (Sue, Sean) conduct assessments. A Role Manager (Sara) oversees the approval stage if the request is successful. This structured use of data and resource allocation ensures a controlled and efficient credit evaluation process.

2 Business Process Model Dependencies

In order to explain the cascading effects of cyber loss events within business processes, this study aims to analyze their impact on the following process model entities: resources, activities, events, and gateways. Our analysis targets following types of dependencies in the process model: *1) Resource-to-Activity:* we investigate how resource participation in a risk event affects the related activities. *2) Control Flow:* we assess the impact of risk events involving activities, events, and gateways on the subsequent activities, events and gateways within the process. *3)Data Flow:* we analyze how the involvement of data resources in a

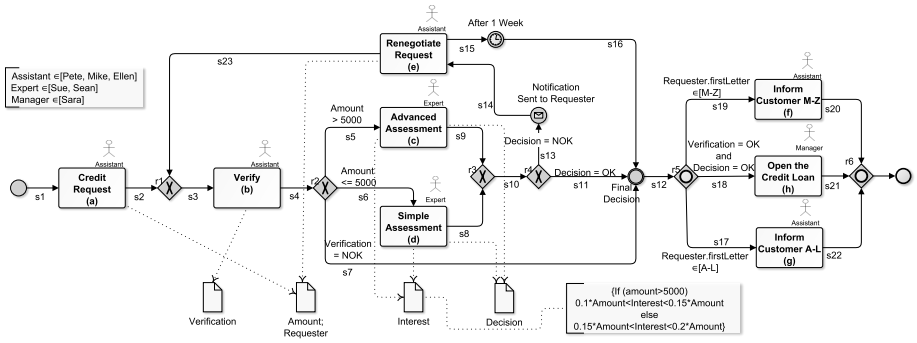

Fig. 1. A data-and-resource-aware process model of a credit evaluation process, as introduced in [2]

risk event influences other entities within the process. Since resource-to-resource dependencies are frequently not explicitly represented in process models and can encompass a broad spectrum of domain-specific dependencies, such as those found in software component relationships [4], we assume that all dependent objects function as input resources for an activity. For instance, when an activity is conducted by a human utilizing a computer, we assume that both the human and the computer are considered resources for that activity. In this section, we provide a precise definition of these types of dependencies based on the BBO model illustrated in Figure **??**. To facilitate this, we utilize formal logic, where predicates represent relationships, and the arguments correspond to entities within the UML model.

Resource-to-Activity Dependencies According to [1], resources within a process are either produced or consumed by activities. A resource is produced when an activity generates outputs, such as data or items, that are necessary for subsequent tasks, for example, creating a report or assembling a product component. Conversely, a resource is consumed when an activity uses it as input, modifying or utilizing it to achieve its objectives, as in the case of utilizing raw materials in manufacturing or processing data for analysis. As noted by [3], the primary dependencies between activities and resources are OR and AND. In an OR dependency, if an activity a_1 depends on resources r_1 or r_2, a_1 can proceed if either r_1, r_2, or both are available. Conversely, in an AND dependency, all required resources must be simultaneously available for the activity to proceed.

In the BBO representation, dependencies between activities and resources are organized into input and output sets. The input set of an activity contains resources necessary to initiate and execute that activity, while the output set includes resources that are generated or modified upon the activity's completion. These resources may include various elements, such as data, documents, personnel, or equipment.

The dependencies of input sets are expressed in equation 1. In our example, the activity required to conduct an *Advanced Assessment* has two primary inputs: a human resource with the role of *Expert* and a data resource representing the credit request *Amount*. All human resources designated as *Experts* are included within a single input set, while the *Amount* data resource constitutes a separate input set. For the activity to proceed, all input sets must be available, reflecting an AND dependency between resources in different input sets, whereas an OR dependency applies within a single input set, indicating that only one resource in that set needs to be available.

$$\begin{aligned}\forall a \forall ios \forall is \forall r \big(& HasIoSpecification(Activity(a), InputOutputSpecification(ios)) \wedge \\ & HasInputSet(InputOutputSpecification(ios), InputSet(is)) \wedge \\ & HasResourceInputRef(InputSet(is), Resource(r)) \rightarrow \\ & InInputSet(r, a) \big)\end{aligned} \tag{1}$$

Similarly, resources in the output set of an activity are represented as shown in 2. For example, after the *Advanced Assessment* activity, output resources might include data elements like *Interest* and *Decision*. Notably, the *Decision* data resource serves as an input for subsequent activities, such as *Renegotiate Request*. These dependencies are formalized in the referenced equation.

$$\begin{aligned}\forall a, \forall ios, \forall os, \forall r \big(& HasIoSpecification(Activity(a), InputOutputSpecification(ios)) \wedge \\ & HasOutputSet(InputOutputSpecification(ios), OutputSet(os)) \wedge \\ & HasResourceOutputRef(OutputSet(os), Resource(r)) \rightarrow \\ & InOutputSet(r, a) \big)\end{aligned} \tag{2}$$

Control Flow Dependencies Control flow dependencies are defining the logical sequence for executing activities in a business process. As [6] describes, control flow patterns fall into two primary categories: synchronized and unsynchronized. Synchronized patterns require multiple process branches to reach specific points simultaneously, ensuring that activities converge before the next stage can proceed. In contrast, unsynchronized patterns allow branches to operate independently, where each branch can advance without needing others to complete. Our study focuses on synchronized control flow patterns, specifically examining the core patterns identified by [6]: Normal Sequence Flow, Exclusive, Inclusive, and Parallel control-flow patterns.

The Normal Sequence Flow pattern is the most straightforward, as it dictates a linear progression where one flow node directly follows another. It is limited to flow nodes, which can be activities or events, distinguishing it from more complex patterns that involve gateways. In our model, these dependencies expressed in equation 3:

$$\begin{aligned}\forall a_1, \forall a_2, \forall s \big(& HasOutgoing(FlowNode(a_1), SequenceFlow(s)) \wedge \\ & HasIncoming(FlowNode(a_2), SequenceFlow(s)) \wedge \\ & (s \in NormalSequenceFlow) \wedge \\ & (a_1 \in \{Event, Activity\}) \wedge (a_2 \in \{Event, Activity\}) \\ & \rightarrow NormalSequenceFlow(a_1, a_2) \big)\end{aligned} \tag{3}$$

The Exclusive pattern selects one path from multiple options based on specific conditions, with only the chosen branch executed. Once it completes, the process resumes at the join without waiting for other branches. The Inclusive pattern allows multiple branches to run simultaneously if conditions are met, pausing at the join until all selected branches finish. The Parallel pattern activates all branches at once, with the process waiting at the join until every branch completes. This pattern is ideal for coordinating parallel tasks that require synchronization.

In our model, these patterns are described as instances of input flow nodes (a_1), and output flow nodes (a_2), connected to the gateway (g) by instances of input sequence flow (s_1) and output sequence flow (s_2), where the output sequence flow might have a conditional expression which routes the control flow according to data resource values. These dependencies expressed in equation 4:

$$\begin{aligned}\forall a_1, \forall a_2, \forall g, \forall s_1, \forall s_2, \forall e \, \big(& HasOutgoing(FlowNode(a_1), SequenceFlow(s_1)) \wedge \\ & HasIncoming(Gateway(g), SequenceFlow(s_1)) \wedge \\ & HasOutgoing(Gateway(g), SequenceFlow(s_2)) \wedge \\ & (g \in \{ExclusiveGateway, InclusiveGateway, ParallelGateway\}) \wedge \\ & HasIncoming(FlowNode(a_2), SequenceFlow(s_2)) \wedge \\ & ((\exists e \, ConditionExpression(SequenceFlow(s_2), Expression(e))) \\ & \rightarrow GatewayPattern(a_1, a_2, g, s_2, e)) \big)\end{aligned} \tag{4}$$

In our case study, the *Verify* activity serves as an instance of the input flow node (a_1) connected by an outgoing sequence flow to an exclusive gateway (g). This gateway has three output flow nodes (a_2)—the activities *Advanced Assessment*, *Simple Assessment*, and the event *Final Decision*. Routing through each sequence flow is guided by its expression (e); for instance, if the amount exceeds 5,000- the *Advanced Assessment* is triggered, and so forth.

Data Flow Dependencies Data flow dependencies illustrate how data resources, such as values, documents, and other data elements, affect and are affected by various process components like activities, events, and routing constraints. [5] conducted an extensive research about the impact of these dependencies in business processes, and identified the following types of dependencies.

Data on Data This dependency describes cases where data resources are interdependent, so that if one data resource is modified, the other is updated accordingly. This is particularly relevant when data values are functions of other data values within the business process. For example, in our case study, the *Interest* value depends on the *Amount* value. This dependency expressed in equation 5.

$$\forall r_1, \forall r_2 \, \big(DependsOn(DataResource(r_1), DataResource(r_2)) \big) \tag{5}$$

Data on Activity & Activity on Data The *Data on Activity* dependency exists when a data resource serves as an input to an activity, influencing its behavior or decision-making process. This is a specific case of the *InInputSet* dependency described in equation 1, where the input resource is of type *DataResource*, as

shown in equation 6. For example, in our scenario, the *Amount* data resource serves as an input for either the *Advanced Assessment* or *Simple Assessment* activities. In contrast, the *Activity on Data* dependency describes instances where an activity modifies a data resource through creation, deletion, or updating. This is a specific instance of the *InOutputSet* dependency outlined in equation 2, with the output resource type as *DataResource*, as expressed in equation 7. For instance, both *Advanced Assessment* and *Simple Assessment* activities create a *Decision* data resource.

$$\forall r, \forall a \left(InInputSet(DataResource(r), Activity(a)) \right) \tag{6}$$

$$\forall r, \forall a \left(InOutputSet(DataResource(r), Activity(a)) \right) \tag{7}$$

Data on Routing Constraint This dependency exists when a data resource is used in an expression that determines the routing of a process case. In our model, this pattern is formalized in Equation 8, represented by a gateway (g) with a conditional sequence flow (s) as an output. The sequence flow (s) has an expression (e) that depends on a data resource (r). In our example, there are two instances of this pattern between the *Amount* data resource and Gateway r2. The first pattern instance is via conditional sequence flow s5, with the expression *Amount* $>$ *5000*, and the second pattern is via conditional sequence flow s6, with the expression *Amount* $<=$ *5000*.

$$\begin{aligned} \forall g, \forall s, \forall r, \forall e \big(& HasOutgoing(Gateway(g), SequenceFlow(s)) \wedge \\ & (s \in ConditionalSequenceFlow) \wedge \\ & ConditionExpression(SequenceFlow(s), Expression(e)) \wedge \\ & DependsOn(Expression(e), DataResource(r)) \\ & \rightarrow DataOnRoutingConstraint(r, g, s, e) \big) \end{aligned} \tag{8}$$

Routing constraint on flow Node This dependency occurs when the activation of a flow node depends on the evaluation of a gateway and a conditional expression that relies on a data resource. In our model, this pattern is formalized in Equation 9 and is represented by a gateway (g) with a conditional sequence flow (s) as an output. The sequence flow (s) contains an expression (e) that depends on a data resource (r) and is connected to a flow node (a). In our example, this pattern exists between the *Advanced Assessment* activity and Gateway r2, via conditional sequence flow s5, with the expression *Amount* $>$ *5000*.

$$\begin{aligned} \forall g, \forall s_1, \forall s_2, \forall r, \forall e, \forall a \big(& HasOutgoing(Gateway(g), SequenceFlow(s_1)) \wedge \\ & (s_1 \in ConditionalSequenceFlow) \wedge \\ & ConditionExpression(SequenceFlow(s_1), Expression(e)) \wedge \\ & DependsOn(Expression(e), DataResource(r)) \wedge \\ & HasIncoming(FlowNode(a), SequenceFlow(s_1)) \\ & \rightarrow RoutingConstraintOnFlowNode(g, a, r, s_1, e) \big) \end{aligned} \tag{9}$$

References

1. Adamo, G., Ghidini, C., Di Francescomarino, C.: What is a process model composed of? a systematic literature review of meta-models in bpm. Software and Systems Modeling **20**(4), 1215–1243 (2021)
2. De Leoni, M., Van Der Aalst, W.M., Van Dongen, B.F.: Data-and resource-aware conformance checking of business processes. In: Business Information Systems: 15th International Conference, BIS 2012, Vilnius, Lithuania, May 21-23, 2012. Proceedings 15. pp. 48–59. Springer (2012)
3. Goethals, F., De Backer, M., Lemahieu, W., Snoeck, M., Vandenbulcke, J., Infrastructures, S.l.E.E.: Identifying dependencies in business processes. In: Communication and Coordination in Business Processes (LAP-CCBP) Workshop, Kiruna, Sweden, June. vol. 22 (2005)
4. Pashchenko, I., Vu, D.L., Massacci, F.: A qualitative study of dependency management and its security implications. In: Proceedings of the 2020 ACM SIGSAC conference on computer and communications security. pp. 1513–1531 (2020)
5. Tsoury, A., Soffer, P., Reinhartz-Berger, I.: Data impact analysis in business processes: Automatic support and practical implications. Business & Information Systems Engineering **62**, 41–60 (2020)
6. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Cham (2012). https://doi.org/10.1007/978-3-642-28616-2