My own process: Providing dedicated views on EPCs

Florian Gottschalk f.gottschalk@tm.tue.nl
Michael Rosemann m.rosemann@qut.edu.au
Wil M.P. van der Aalst w.m.p.v.d.aalst@tm.tue.nl

Abstract: The idea of Business Process Management demands that companies change their focus from optimising tasks to focusing on whole business processes optimising the overall value chain. However process models depicting such complex processes are perceived as complicated and therefore as hard to use. The critical task is to present only relevant model parts to users and at the same time enable them to locate their contribution within the entire value chain.

This paper discusses an approach for tailoring Event-driven Process Chains to those parts that are relevant to selected organisational units. The approach uses the allocation of organisational units to functions as a selection criteria for relevant model parts. A distinction between concurrent and alternative collaboration and the implementation of a corresponding notation within the EPC notation enable the introduction of additional process interfaces, a standard feature of Event-driven Process Chains, into the tailored models. The process interfaces ensure the visibility of the connected business process. Therefore the approach helps to resolve the depicted conflict.

1 Introduction

In order to handle and accurately describe business processes for all parties involved, today's companies are using numerous modelling techniques, each aiming at different goals and audiences. This leads to a significant complexity of the modelling landscape. A high degree of complexity however results into a decrease of user acceptance [RSD05]. For that reason the quality of conceptual models is subject of academic research for a long time (e.g. [LSS94, Ros96]). When creating models, companies have to incorporate the same factors as for every other product, i.e. time, costs, and quality. The impact of these factors also depends on the purpose of modelling. E.g., a model created for simulation purposes will differ from the model created for knowledge management or organisational documentation. Also the HR manager's demand on models of the company will for sure differ from the requirements of technicians. Proces modelling for various user groups or purposes is called *multi-perspective modelling* whereas each perspective is a subset of the total model [Ros03].

Powerful tools like the Architecture of Integrated Information Systems (ARIS) [Sch94b, Sch00] support the creation of such multi-perspective models. The architecture enables the integration of different perspectives. It distinguishes between an object and its oc-

currences. By using occurrences, the same object can be used in several models and modelling perspectives. Within ARIS, the process modelling language of *Event-driven Process Chains* (EPCs) is used as an anchor point for the integration of the different perspectives. EPCs are commonly used to depict the control flow of a business process, i.e. the order in which tasks have to be performed (see Figure 1). In addition EPCs provide the integration of multiple perspectives. They allow connecting occurrences of elements used within specialised perspectives (e.g. data, organisational units, or utilities) to functions. So the relevance of the particular element for the function becomes obvious. However, the current assignment notation lacks of information about the interaction between multiple connected elements. For that reason we introduce new connectors which are depicting different kinds of collaboration in the first part of this paper (e.g., see function *Release Invoice manually* in Figure 1). We focus on the assignment of organisational units to functions, but connectors for other assignments should be definable in a similar way.

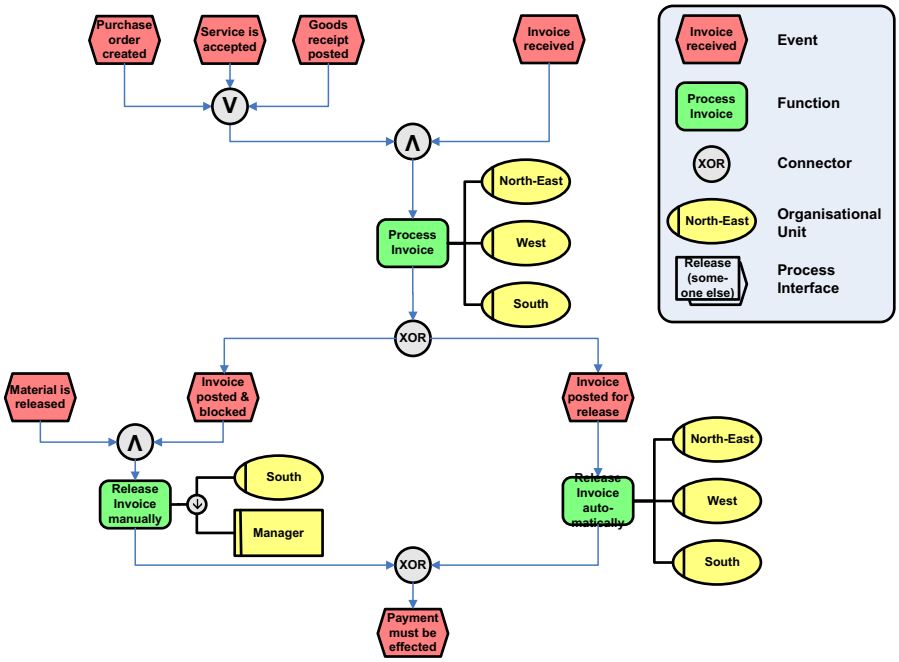

Figure 1: Example Invoice Verification Process

By adding additional elements like stakeholders or data to the process flow even small EPCs can become fairly complex [KKS04]. E.g., the simple and clearly arranged invoice verification process in Figure 1 needs almost half an A4 page. Also the structure of EPCs (with, functions, surrounding events, connectors, and arcs in between) drives the model complexity, especially when creating models with a large number of functions and connectors. However, it is important that a user of a model quickly identifies those parts of the

model that are relevant to him, i.e. the level of information presented must correspond to his requirements [MG75, BDFK03]. Thus, we introduce a reduction mechanism for EPCs in the second part of the paper so that the resulting process model is of high relevance for a selected organisational unit. E.g., in the depicted invoice verification process the manager should just see the function *Release Invoice automatically* and its direct environment. For this we not only consider the selected elements from the original process (similar to [BDFK03, BDKK02, RSD05]) but also introduce interfaces making the overall process flow and therefore the contribution to the value chain visible.

To conclude we summarise the contribution of this paper and we give an outlook on potential future extensions.

2 Assigning Organisational Units to Functions: Who has to do it?

To depict the involvement of organisational units within a process, EPCs allow connecting organisational units to functions. The reasons for such a connection (and therefore for the involvement) are manifold. E.g., the ARIS Toolset [Sch94a] suggests reasons like:

- The organisational unit executes the function.
- The organisational unit contributes to the function.
- The organisational unit must be informed about result of the function.
- The organisational unit has a consulting role in the function.

In the definition of eEPCs it is requested that each organisational unit is involved in at least one function and that each function is allocated to at least one organisational unit as depicted in the meta model in Figure 2.

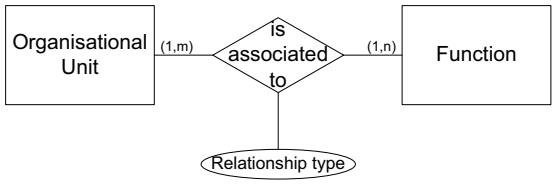

Figure 2: Meta model for allocating organisational units and functions (adapted from [RzM97, Sch00])

The meta model depicts that an organisational unit can be involved in more than one function as well as more than one organisational unit can be involved in a function – also for the same reason. E.g., it may be required that the management as well as the sales department have to contribute actively to the performance of a market analysis. Table 1 provides a matrix illustration of the involvement of different departments in a real-world order processing situation where three kinds of involvement are distinguished [Sch00]. The notion of the matrix quasi conforms to so-called RACI matrices [PW05], just the labeling differs.

Organizational Units / Functions	Management	Marketing	Management & Organization	Branch Management	HR	Cost Acc. & Controlling	Sales	Sales and Distribution	R&D	Production	Purchasing / Procurement	Materials Warehouse
Market Analysis	i	r	a	a			i					
Production Program Planing	r	i	i	a	a	a	i		a	a	a	
Proposal Processing							r					
Order Processing							r					
Product Development	i			a	a	i	i		r	i	i	
Production Planing			i		i	a	i		i	r	i	a
Materials Purchasing											r	i
Warehouse Management								a				r
Prroduction Mgt. & Control						a	a					
Quality Assurance				a			a	a	i	r	i	r
Shipping				i			i	r				
Cost Accounting & Control	a		a	i	a	r			a	a	a	
Financial & Investment Plan.	i		r	a	a	i				a	a	
HR Planning & Development	i		i	a	r	a				a	a	
Inventory & Year-end Closing	i		a	i		r				a	a	

r = responsible i= actively involved a=associated

Table 1: Matrix illustration of a function allocation [Sch00]

Neither the matrix nor a corresponding EPC depicts interrelations between different organisational units involved in the same way to the same function. E.g., it might be possible that not both management and sales are required but rather each of them is able to perform a market analysis. That means several organisational units can be involved in a function concurrently or alternatively. Obviously this makes not only a huge difference for ressource utilisation, but also influences the synchronisation of the individual user tasks. So the opportunity to specify this accurately should be provided by the modelling environment.[1]

[1]The simulation of the ARIS Toolset offers the opportunity to specify if allocated organisational units are involved either concurrently or alternatively by maintaining Boolean attributes. It is not possible to specify a combination of concurrent and alternative involvement in the same function nor to specify different types of resource allocation for different reasons. Both might be required if more than two organisational units can be or are involved in a function.

Hence we suggest to use a logical term for specifying such relationships. E.g., the active involvement for market analysis in Table 1 can be depicted as *Management* AND *Sales* for concurrent involvement or as *Management* XOR *Sales* for alternative involvement. In product development more than two organisational units are actively involved, e.g., the relationship might be *Purchasing / Procurement* AND *Cost Acc. & Controlling* AND (*Management* XOR *Production* AND *Sales*). Within this notation it is assumed that precedence is the same as in Boolean algebra or common programming languages, i.e. the precedence of the AND operator is higher than the precedence of the XOR operator. The notation also allows depicting relationships like "two out of three associated organisational units are required". E.g., if *A*, *B*, *C* are the possible organisational units the term would be *A* AND *B* XOR *A* AND *C* XOR *B* AND *C*.

To depict such relationships within an EPC process model, all terms must be expanded until brackets are not required anymore. E.g., for the active involvement in product development that would result in the association *Purchasing / Procurement* AND *Cost Acc. & Controlling* AND *Management* XOR *Purchasing / Procurement* AND *Cost Acc. & Controlling* AND *Production* AND *Sales*. This distinguishes clearly both possible involvement combinations. It allows grouping all organisational units that are required within each combination by an AND-connector. Afterwards the AND-connectors can be connected to the function. Then all connections of organisational unit groups to the function are the alternatively involved organisational unit groups. The resulting model for this example is displayed in Figure 3.

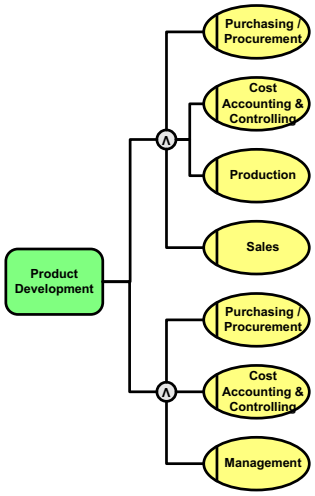

Figure 3: Multiple Organisational Units associated to a function in two alternative groups

In addition to the distinction between alternative and concurrent involvement, the concurent involvement can be further specified. E.g., it makes a difference for involved organisational units, if the task is a meeting where all organisational units have to perform the

task together at the same time, or if each organisational unit gives its own contribution to a function that can be executed in parallel to and independent of other organisational units. A third option would be that the organisational units are involved in a certain order. In general, it is possible to distinguish between *joined*, *parallel*, and *sequential* involvement. To make this distinction visible, the AND-connector may be converted to specialised connectors depicting the particular relationship. E.g., the joined involvement can be depicted by the AND-symbol, the parallel involvement by two vertical lines, and the sequential involvement by an arrow showing the order of involvement as depicted in Figure 4.

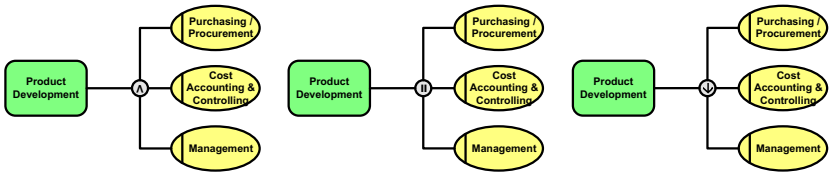

Figure 4: Possible illustration of joined, parallel, and sequential involvement

Taking Guidelines of Modelling [Ros96, BRS95] into consideration, such new, specialised symbols can be put into question. E.g., the collaboration could also be specified through the process flow in a hierarchical lower level EPC. In fact, it depends on the purpose of modelling if the relationship should be depicted within the model, maintained in attributes, and/or specified on a lower level EPC. When making this decision the matching of levels between the organisational hierarchy and the process model hierarchy must be taken into consideration as well.

To depict grouped association between organisational units and functions within the meta model the entity type *organisational unit grouping* must be introduced. The entity type can represent either a joined, a parallel, or a sequential collaboration. In case of a sequential collaboration also the position of the organisational unit within the sequence must be maintained. This is done in the association between the *organisational unit* and the *organisational unit group* (see Figure 5).

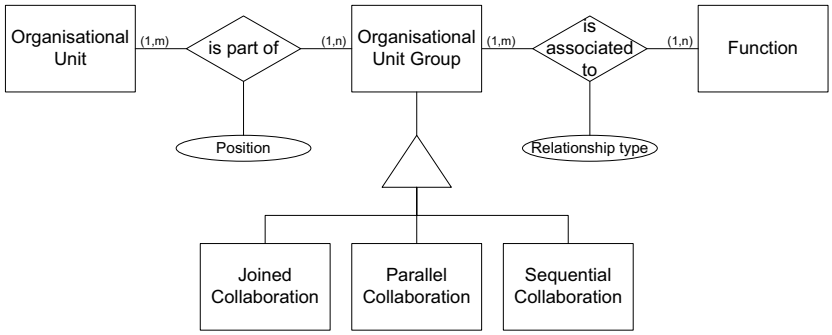

Figure 5: Meta model for allocating organisational units to functions via organisational unit groups

3 Deriving Individual Process Models: What do I have to do?

Handling of complexity and adjusting the scope of models to the requirements of the user is of prime importance to facilitate the usage of models ("Guideline of relevance" [Ros96, BRS95]). The allocation of organisational units to functions can be used as a selection criteria. It enables hiding of unselected process elements which are irrelevant to the particular organisational unit. I.e. the complex model is reduced to clearer, smaller models and the model's degree of relevance increases (see [BDFK03]). However, in this case the user is not aware of the overall business process as all visibility of process elements in which the particular organisational unit is not involved in is removed. Such an approach would conflict with the idea of Business Process Management that demands comprehensive business processes addressing the whole value chain and not ending at the borders of an organisational unit [HC93].

Indeed we argue in our approach that functions in which the considered organisational unit is not involved are irrelevant and remove them from the models. Nonetheless it is important to depict the link to these functions so that the business process flow is kept visible. Consequently, our approach uses the concept of process interfaces within EPCs. Process interfaces are navigation aids that show the link from one to another process [KT98]. They visualise the connection between processes on the same hierarchical level [KKS04]. I.e. in our approach we introduce a process interface to the process model whenever other organisational units are involved in the same, preceding or succeeding task of a business process. So the process interfaces keep the inter-organisational unit process flow visible although irrelevant parts are not included in the specific model. E.g., Figure 6 depicts the invoice verification process from Figure 1 tailored for the manager. He is only confronted with

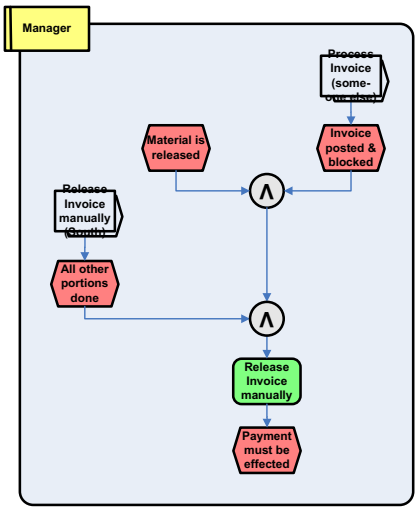

Figure 6: Manager's Invoice Verification Process

those parts of the process he is involved in and the process interfaces clearly depict what has to be done before he becomes involved. So the manager is aware of his contribution within the overall process flow but he is not confronted with irrelevant model parts.

The developed algorithm can be divided into five main steps:

1. Copy[2] all functions performable by the considered organisational unit into a new individual process model.
2. Copy for all these functions their preceding and subsequent (start- and end-) events as well as the in-between connectors and arcs from the original process model to the individual process model.
3. Analyse for each function all preceding and subsequent functions regarding their possible alternative executing organisational units. If required, introduce new process interfaces and events and integrate them into the individual process model.
4. Analyse for each function if it must be executed together or in parallel with other organisational units. If required, introduce new process interfaces and events and integrate them into the individual process model.
5. Remove unnecessary connectors from the individual process model.

The first step ensures that all functions that can be performed by the particular organisational unit appear in the individual process model as well. The performable functions can easily be derived from the EPC by analysing the connections of organisational units to functions. The second step forms an EPC around the copied functions by re-establishing its original environment. Thereby, and (if not otherwise depicted) in the following description of the algorithm the term *copy* is meant in the sense of "copy this occurrence and replace it if already existing". That means the same occurrence of an event also occurs in the individual model only once.

These two steps already derive a syntactically correct EPC that includes all executed functions and that conforms to the model derived by the element selection of [BDFK03]. However, the derived process models reflect only the parts of the process executed by the considered organisational unit. To keep the visibility of the overall business process, organisational breaks have to be analysed and additional entrance and exit points to and from the process have to be introduced as process interfaces. Such process interfaces will

[2] In fact, the algorithm does not hide non-required elements as suggested in [BDFK03]. Instead it copies occurrences of the required elements and adds the process interfaces to a new process model. This is done to address some further demands on tools for comprehensive process modelling. Among other requirements such a tool must allow maintaining and managing of all different modelling phases [BDKK02]. It might be required that a derived individual model for an organisational unit has to be modified, i.e. different organisational units end up in different process models [RA05, DCRvdA05]. If this would be done on the same model with hidden elements, each decision would effect the process within other organisational units. A high degree of standardisation and equal processes among different organisational units is indeed desirable, but it is not always practicable (e.g., because of different legal regulations in different countries). That means such a standardisation of individual processes for organisational units requires the consolidation of the individual process models back into a comprehensive business process model. However, this consolidation is out of the scope of this research.

depict that other organisational units are executing functions which are connected to the process of the particular organisational unit.

For this purpose it is required to analyse all preceding and all subsequent functions (below also called *connected functions* or *A*) for each of the copied functions (below also called *original function* or *B*) regarding the organisational units performing them (Figure 7). If a function is analysed by the algorithm, this reflects the runtime situation that the particular organisational unit performs this function. Thus it is irrelevant which other organisational units can perform the considered function itself alternatively. It is also sufficient to analyse the performance of directly preceding and succeeding functions. If these connected functions can be performed by the particular organisational unit (and therefore are relevant), they will be analysed by the algorithm on their own.

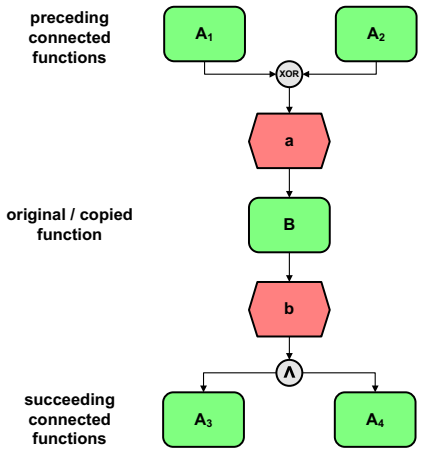

Figure 7: Preceding and succeeding functions

When regarding the groups of organisational units that are performing the connected functions, we have to distinguish between three types of groups:

- Only groups of organisational units to which the regarded organisational unit belongs can perform the function: That means the particular organisational unit has to perform this function in any case.
- Only groups of organisational units to which the organisational unit does not belong can perform the function: That means the particular organisational unit must not perform the function. As the regarded organisational unit is not involved in this connected function, it is irrelevant which other groups of organisational units are able to perform that function: The own process starts or stops at the event.
- Groups of organisational units to which the regarded organisational unit belongs as well as groups of organisational units to which the regarded organisational unit does not belong can perform the function: If in the particular instance of the process the

organisational unit is not involved in the connected function, it is irrelevant to the particular function which organisational unit group is involved as the own process starts or stops at the event.

Two functions within an EPC must always be connected via an event in between. In addition to the event, also one or more connectors can be located in between. However this has no influence on our approach and can therefore be neglected.

As some functions require not only a single organisational unit but whole groups of organisational unit for their execution, also the synchronisation of the individual processes for the involved organisational units must be ensured. This is done in the fourth algorithm step. As depicted in section 2 we have to distinguish if both organisational units must perform the functions together, in parallel or in a sequential order. Executing the function together means that all organisational units must perform the function at the same time, whereas executing the function in parallel means that the function is only completed if all involved organisational units have performed and finished their portion. Performing the function in a sequential order means that the required organisational units neither can perform the function at the same time nor they need to perform it together. They have to execute it one after another.

The final step is a cleanup-step. Some connectors copied by the algorithm will be connected to only two other elements. These connectors are not required and should be replaced with shortcuts.

The first two steps as well as the fifth step do not distinguish different scenarios depending on the context. However, the results of the third and fourth step depend on the context of the particular function. Therefore we will now analyse how the different situations can be handled. Afterwards we will provide a small example application.

3.1 Analysis of Scenarios for Alternative Execution

The analysis of Scenarios for alternative execution is grouped by the relation between the considered organisational unit and the organsiational units involved in the connected function:

Function must be executed by the same organisational unit

Obviously, if the preceding, connected function A must be executed by the same organisational unit as the regarded function B (see Figure 8), A has already been copied to the new process in the first algorithm step. Both functions ensure that the event in between is copied as well. Each function ensures that all connectors between the particular function and the event are copied. As the process flow between both functions does definitely not change the executing organisational unit no process interfaces to other organisational units have to be introduced. That means this situation does not require any further treatment in this algorithm step. Of course, the same argumentation will hold for the other way around,

i.e. if A would be the regarded function and B would be connected to it.

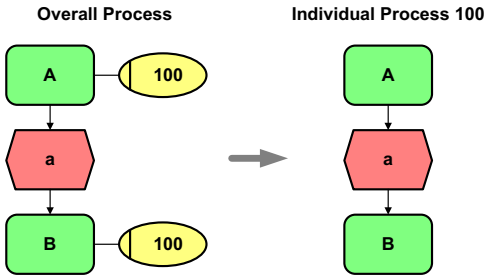

Figure 8: Functions executed by the same organisational unit

Function must be executed by one or more other organisational units

If a preceding function A has to be executed by one or more other organisational units this should be depicted in the individual process by replacing the function with a process interface. To insert this process interface it should be named like the function with the addition that it is performed by another organisation unit. To connect the process interface all arcs and connectors between the already copied event and the preceding function A (within the original process model) must be copied to the individual process – however the connection to A in the original model must be connected to the process interface instead. Connectors in between have no influence on the copy process. If connectors in between exist they are just copied into the resulting process as in Figure 9. However, only the arcs on the path between the function/process interface and the event are copied. If other arcs are required, they will be copied when regarding the particular connected function.

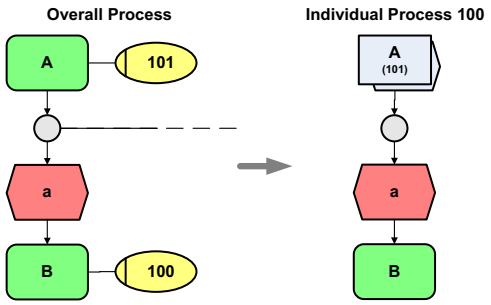

Figure 9: Preceding function, executed by another organisational unit

The process interface enables a clear identification of what has been done before the individual process starts as well as why it starts. It allows keeping track of the business

process. Also in this case the same argumentation would hold for a succeeding function. A separate description is therefore omitted.

Function can be executed by the same or other organisational units alternatively

If a function can be executed by the same as well as by other organisational units, we need to distinguish between preceding and succeeding connected functions. Of course, as above, if a preceding function A is executed by one or more other organisational units this should be depicted in the individual process as a process interface. However, in this scenario it is also possible that A is executed by the organisational unit itself (or a group of organisational units where it belongs to). For that reason the process interface must be introduced in addition to the function A which was already copied in step one of the algorithm. In step two the connection path between the function A and its end-event a was already copied to the individual process. To integrate the new process interface, the arc which is connecting A with the first succeeding connector or, if no connector exists, the arc which is connecting A with its end-event a must be removed. Instead of the arc function A and the new process interface must be connected to a new XOR-join-connector. Then the XOR-connector is connected to the element to which A was previously connected to (see Figure 10). As the arc that was connected directly to function A is broken up, connectors between A and event a have no further influence on the algorithm.

The introduced process interface depicts a new starting point of the process for the case that the regarded organisational unit does not contribute to function A. At the same time the opportunity is kept that the regarded organisational unit contributes to both functions.

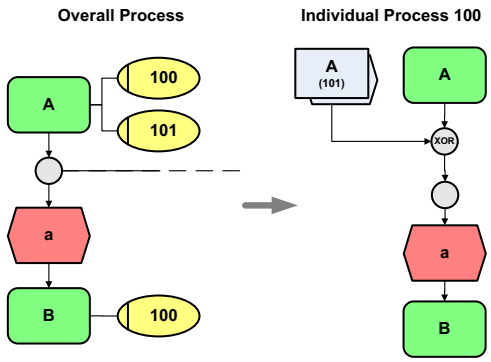

Figure 10: Preceding function, executed by the same or another organisational unit alternatively

The succeeding scenario is a bit more difficult to handle as one out of at least two organisational units (or groups) has to continue executing the process and a decision must be made who continues. Either the particular organisational unit can continue executing the process by performing the next function or the process is handed over to another organisational unit that is allowed to perform the succeeding function. Obviously the distinction between these two possibilities must be depicted as an XOR-split-connector in the individ-

ual process model. Theoretically, three options exist to introduce this connector. First of all, according to the cases above it could be introduced directly before function A. However, this would lead to an incorrect EPC as a split connector would follow an event and an event is not able to perform any decision. The second option would be to introduce the XOR-split-connector directly before event b. This would lead to a syntactically correct EPC. However, it implicates some semantic problems. Firstly, it adds additional functionality to the predefined function B. This could cause problems if a further specifying process model is assigned to the function that does not include the decision. Additionally, this solution will cause huge efforts if b is followed by an AND-split. According to the EPC rules this is permitted. However, introducing the XOR-split before b would require copying the AND-split as well. All decisions made for other connected functions to the end split (that are also analysed as they are connected to B as well) must then be transferred to the new branch in addition.

The third and preferred possibility is to introduce the decision task as a separate function that is followed by the XOR-split-connector with the end-events *proceed* and *other organisational unit proceeds*. The event *other organisational unit proceeds* is connected to the new process interface for depicting that in this case the process continues, but is not executed by this organisational unit. For resulting into a valid EPC the whole construct is inserted into the process directly before function A and behind any connector (see Figure 11). This solution does not add any additional functionality to a function. However, it involves a certain risk for overemphasising the decision task as it is depicted in the same way as other tasks. In addition, the function may lead to the assumption that a free choice for the employee will exist if he/she continues, whereas in practice probably a strict set of rules will exist. Thus the decision function is nothing else than applying this set of

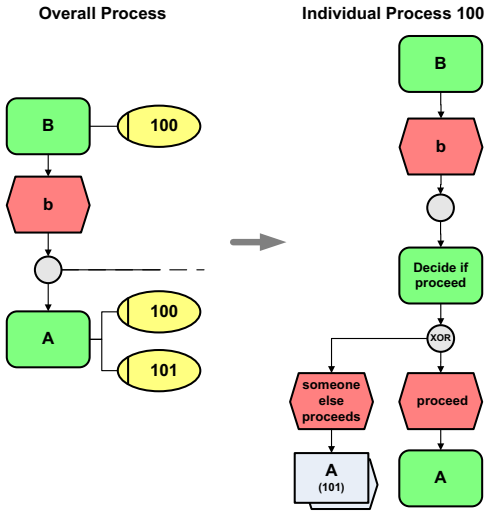

Figure 11: Succeeding function, executed by the same or another organisational unit alternatively

rules to the actual process. To handle pools of tasks, it is required to allow an implicit performance of the decision function. If more than one organisational unit including the regarded unit is allowed to execute function A, all of them will use the same in-tray that is at the same time the out-tray of function B. If function B was executed by the regarded organisational unit, and another organisational unit picks up the process to continue, the regarded organisational unit implicitly decides not to continue. To legitimate this form of decision making, e.g., it is possible to say the organisational unit made the decision by not continuing directly or by waiting too long before continuing.

3.2 Analysis of Scenarios for Concurrent Execution

After introducing new process interfaces for alternative executing organisational units, it must be analysed if additional organisational units are required to execute functions. Thus, each function executed by the regarded organisational unit must be analysed regarding additionally required organisational units. If additional organisational units are required, a synchronisation of the processes must be performed as follows.

Parallel performance

If a function must be performed by two or more organisational units in parallel – as depicted above that means that each organisational unit can start and perform its portion of work individually as soon as the function's start event has occurred – no organisational unit will have to wait for others. However, for completing the function and firing its end-event all organisational units involved must have finished their tasks. Thus, the synchronisation of the two (or more) processes should be performed directly after the function and therefore before all possible end-events. The synchronisation can be depicted as an AND-join-connector that fires the subsequent process flow only if all preceding paths are fired. It has to merge a new additional process interface with the organisational unit's process flow and should be named like the function executed in parallel added by a comment that this is the part of the function all others involved perform (see Figure 12). As the synchronising

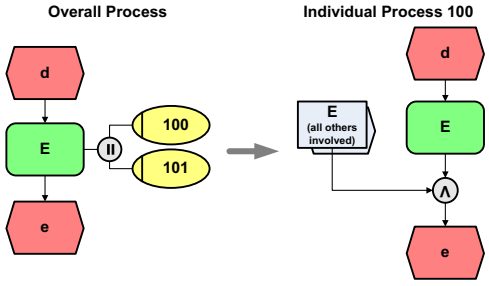

Figure 12: Parallel execution

join is introduced directly after the function, further connectors have no influence on this scenario.

Joined performance

If a function must be performed by several organisational units together, e.g., if the function is a meeting, no organisational unit can start the function without the others. For that reason the synchronisation must already be done before the start of the function. Thus introducing the AND-join-connector is required directly before the function. As according to the eEPC-rules it is not allowed that a process interface is directly followed by a function, an additional event must be introduced between the synchronisation connector and the also newly introduced process interface. The process interface depicts the previous tasks of all others involved executing the particular function and should be named accordingly. The event depicts that all others have finished their previous tasks and are ready to start performing the particular function and should be named accordingly as well (see Figure 13).

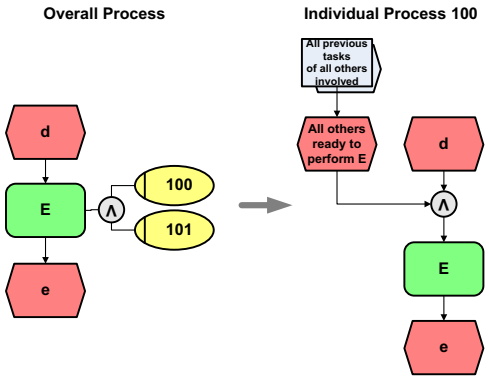

Figure 13: Joined execution

Sequential performance

The third possible scenario for multiple organisational units involved in the execution of a function is that the organisational units execute the function in a sequential order. E.g., it might be required that a manager signs the invoice release after it was processed by a clerical assistant. Probably this situation would be most obvious if the particular parts are modelled by separate functions, each executed by the particular organisational unit. However, if the individually tasks are joined into one function, a synchronisation must be introduced. The synchronisation depends on the position of the particular task within the function. The contribution of the regarded organisational unit can be the first to be executed, it can be executed in between the contributions of others, or it can be the last to

be executed.

If the organisational unit contributes first, the function in the individual process can start directly after the previous event has occurred, but the end-event cannot occur before all others have finished their contribution. Thus a synchronisation must take place before the end-event occurs – identically like in the parallel execution scenario. If the organisational unit contributes last, it cannot start performing the function before all others have finished their portion. A synchronisation with the others has to be performed before the regarded organisational unit starts – similar to the joined execution scenario. In this scenario, however, the process interface should be called according to the regarded function.

If the regarded organisational unit has to contribute in between, obviously it has to wait until other organisational units have finished their tasks. After performing the task, again some other organisational units have to contribute before the end-event of the function signalises the completion of the function (see Figure 14). Thus, this situation is a combination of the two other scenarios – or more correct, the two scenarios above are just specialised scenarios in which no others have to perform tasks before or afterwards.

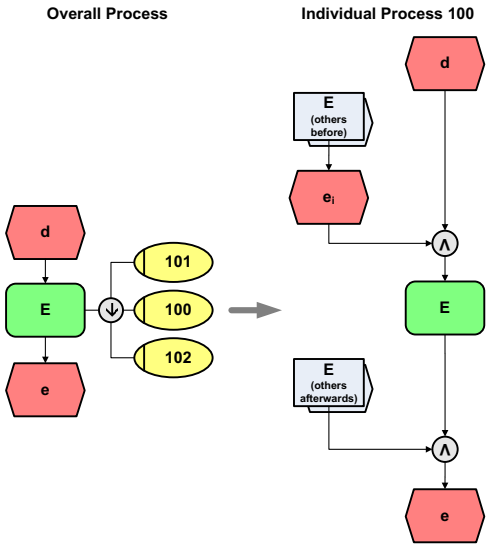

Figure 14: Sequential execution

3.3 Example

To conclude this section we will give a short example how individual process models can be derived. Therefore we use the simple invoice verification process from Figure 1. First, one of the three Organisational Units *North-East*, *West*, or *South* has to process the

invoice. Each of these organisational units can also perform a automatic invoice release. However, a manual invoice release can only be performed by the organisational unit *South*. Afterwards also a Manager has to approve the manual invoice release.

Exemplary the derived individual processes for *West* is depicted in Figure 15. *West* can process the invoice verification and the automatic invoice release, i.e. the particular functions and the surrounding events as well as arcs and connectors in between are copied to the individual process. If the invoice requires a manual release, it must be handed over to *South*, depicted by a new process interface. The function *Process Invoice* can also be executed by *North-East* and *South*, i.e. *West* must be able to pick up the process from these organisational units in order to release it automatically. This is depicted by the process interface in parallel to the *Process Invoice*-function. The two other organisational units are also able to perform the automatic release. That means, after processing the invoice *West* has to decide if it continues with the automatic release or if it leaves that task to one of the other organisational units. Also note that unnecessary connectors as e.g. the one between the function *Relesase Invoice automatically* and the event *Payment must be effected* are removed from the individual model.

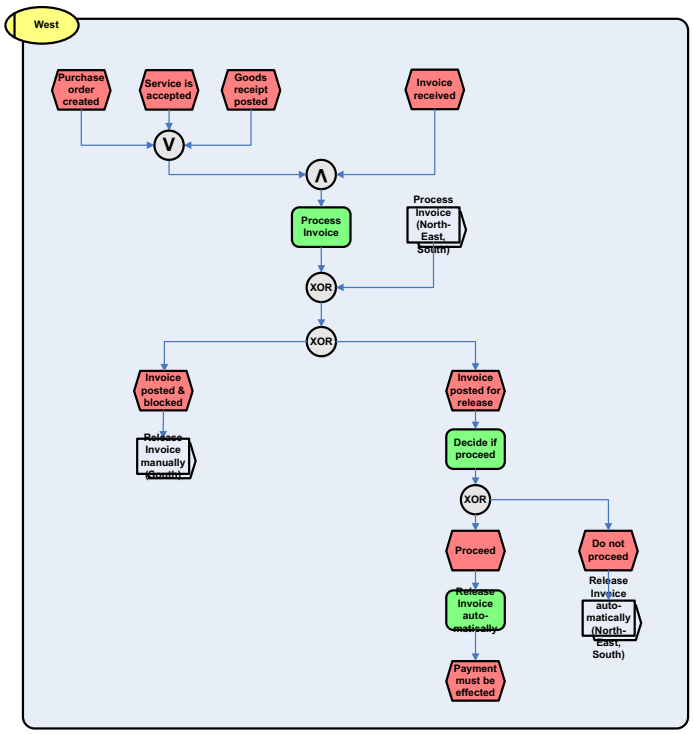

Figure 15: Individual Processes for the Organisational Unit *West*

The manager's process was already depicted in Figure 6. His only task is to agree to the manual invoice release. Therefore he has to wait for processed invoices requiring a manual release, which is depicted by the upper process interface in its individual process. However, he can only agree to the manual release after *South* has performed its contribution to the function. This is also depicted in the process by the second added process interface.

4 Summary and Outlook

Multi-perspective modelling results into fairly complex models although for each user only a subset of these models is relevant. Tailoring mechanisms hiding irrelevant model parts were therefore developed in previous research. These approaches however lack of visibility of the overall business process which is one of the most important ideas behind business process management and modelling. With process interfaces EPCs already provide a construct for keeping this overall process awareness.

Within our approach, we provide a mechanism to introduce such process interfaces quasi automatically. Besides regarding organisational breaks it is also required to distinguish between alternative and concurrent collaboration. If it is possible that a task can be performed by organsiational units alternatively, it must be analysed if additional organsiational breaks can occur. If it is required that organisational units perform a task together their processes must be synchronised. We even distinguished between joined, parallel, and sequential collaboration in a task and also provided a notation to depict these collaborations in general EPCs.

Of course the described approach is limited in several aspects which have to be considered in further research. As already depicted we focus on the involvement of organisational units within a process. However individual models might be interesting for other aspects as well, e.g. for products or locations. Theoretically individual models could be created for every attribute assigned to functions. Further on the approach neglects the hierarchy between organisational units. E.g. an individual process model for an organisational unit should probably include all functions that are assigned to its subordinated organisational units. It might also be interesting to analyse interdependencies between such hierarchies and the introduced notion of collaboration. Also dependencies between assignments of organisational units to functions within a single process instance are neglected, i.e. it might be required that an organisational unit performing a certain function also performs other functions in the same process instance. E.g. the ARIS Process Performance Manager offers the opportunity to specify the release of used resources. The consideration of such dependencies would reduce the individual model size and increase the relevance of the model even further. On the other hand it might also result into multiple versions of the same process for the same organisational unit (which creates further demands on modelling tools). Overall the suggested approach results into an increased complexity of the model base. But it reduces the complexity of the models presented to users and therefore facilitates their usability.

References

[BDFK03] J. Becker, P. Delfmann, T. Falk, and R. Knackstedt. Multiperspektivische ereignisgesteuerte Prozessketten (in German). In M. Nüttgens and F.J. Rump, editors, *EPK 2003: Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, pages 45–60, Bamberg, October 2003.

[BDKK02] J. Becker, P. Delfmann, R. Knackstedt, and D. Kuropka. Konfigurative Referenzmodellierung (in German). In J. Becker and R. Knackstedt, editors, *Wissensmanagement mit Referenzmodellen. Konzepte für die Anwendungssystem- und Organisationsgestaltung*, pages 25–144. Heidelberg, 2002.

[BRS95] J. Becker, M. Rosemann, and R. Schütte. Grundsätze ordnungsmäßiger Modellierung (in German). *Wirtschaftsinformatik*, 37(5):435–445, 1995.

[DCRvdA05] A. Dreiling, M. Chiang, M. Rosemann, and W. M. P. van der Aalst. Towards an Understanding of Model-Driven Process Configuration and its Support at Large. In *Eleventh Americas Conference on Information Systems*, pages 2084–2092, Omaha, NE, 2005.

[HC93] M. Hammer and J. Champy. *Reengineering the corporation: A manifesto for business revolution*. Nicholas Brealey Publishing Allen & Unwin, London St Leonards, N.S.W, 1993.

[KKS04] R. Klein, F. Kupsch, and A.-W. Scheer. Modellierung inter-organisationaler Prozesse mit Ereignisgesteuerten Prozessketten (in German). Technical Report 178, Institut für Wirtschaftsinformatik, Saarbrücken, November 2004.

[KT98] G. Keller and T. Teufel. *SAP R/3 Process-oriented Implementation: Iterative Process Prototyping*. Addison Wesley Longman, Harlow, England Reading, Ma., 1998.

[LSS94] O. I. Lindland, G. Sindre, and A. Solvberg. Understanding Quality in Conceptual Modeling. *IEEE Software*, 11(2):42–49, March 1994.

[MG75] D. Miller and L. A. Gordon. Conceptual Levels and the Design of Accounting Information Systems. *Decision Sciences*, 6:259–269, April 1975.

[PW05] M. Price and J. Works. Balancing Roles and Responsibilities in Six Sigma. 2005. http://finance.isixsigma.com/library/content/c040211a.asp.

[RA05] M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 2005. (to appear, also available via BPMCenter.org).

[Ros96] M. Rosemann. *Komplexitätsmanagement in Prozessmodellen: methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung (in German)*. Wiesbaden, 1996.

[Ros03] M. Rosemann. Preparation of process modeling. In J. Becker, M. Kugeler, and M. Rosemann, editors, *Process Management: A Guide for the Design of Business Processes*, pages 41–78. Springer Verlag, Berlin, Heidelberg, New York, 2003.

[RSD05] M. Rosemann, A. Schwegmann, and P. Delfmann. *Vorbereitung der Prozessmodellierung (in German)*, pages 50–107. Springer, 2005.

[RzM97] M. Rosemann and M. zur Mühlen. Modellierung der Aufbauorganisation in Workflow-Management-Systemen: Kritische Bestandsaufnahme und Gestaltungsvorschläge (in German). In S. Jablonski, editor, *EMISA-Fachgruppentreffen 1997*, pages 100–118, Darmstadt, 1997.

[Sch94a] A.-W. Scheer. ARIS Toolset: A software product is born. *Information Systems*, 19(8):607–624, December 1994.

[Sch94b] A.-W. Scheer. *Business Process Engineering, Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.

[Sch00] A.-W. Scheer. *ARIS - Business Process Modeling*. Springer, Berlin New York, 3rd edition, 2000.