From Process Models to Business Landscapes

Oliver Kopp, Hanna Eberle, Tobias Unger, Frank Leymann
University of Stuttgart, Institute of Architecture of Application Systems
{kopp, eberle, unger, leymann}@iaas.uni-stuttgart.de

Abstract: Today, architecture and business processes are modeled separately. The only integration in architectural diagrams is done with Petri nets in the Fundamental Modeling Concept. Since business users prefer EPCs over Petri nets, we show how information of extended EPCs can be transformed into business landscapes. This facilitates development of IT landscapes satisfying the requirements of the business process and adoption of existing IT infrastructures to new requirements.

1 Introduction and Motivation

Business process models show how business functions relate to each other in a concrete scenario. Business landscapes provide a global view on all business functions and business items within a company. So far, business landscapes and business processes have been modeled separately. The automatic inclusion of the information of business processes in business landscapes has been done manually. In this paper, we present an approach to map the content of business processes to business landscapes. This enables a top-down approach for modeling IT infrastructures: First, the business process is modeled. Second, the presented mapping is used to automatically derive a business landscape. Afterwards, this landscape serves as basis for creating an IT landscape supporting the modeled business process.

Another application scenario is to identify deficiencies in an existing IT infrastructure with respect to a specific business model as illustrated in Figure 1. Using our approach, a business landscape is generated. The elements of this landscape can then be placed in an existing (pure) IT landscape. The result is a combined landscape, which contains both IT and business aspects. The result of the placement helps to identify lacks in the existing infrastructure. For example, the fact that some business items cannot be placed is an indicator that the IT architecture does not fully support the business process.

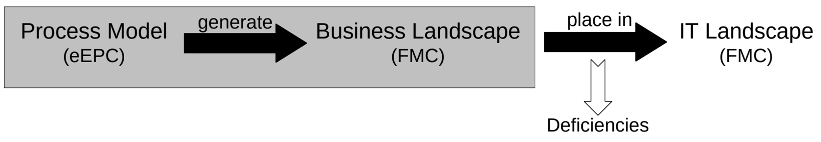


Figure 1: Approach overview

As notation for business processes, we use extended Event-driven Process Chains, which are briefly explained in section 2. We chose the Fundamental Modeling Concepts as notation for business landscapes and present the basics in section 3. The main section 4 is devoted to the mapping of extended Event-driven Process Chains to business landscapes, covering in detail how functions, organizational units and data are mapped.

2 Event-driven Process Chains

We use extended Event-driven Process Chains (eEPCs) as the input for the mapping. eEPCs are Event-driven Process Chains (EPCs, [KNS92]), where functions can be annotated with elements (cf. [STA05], where the usage of eEPCs is illustrated). For our mapping, we restrict the elements to "organizational unit" and "information item." An organizational unit may be connected with the "is involved with" relation. This relation combines all relations of organizational units with functions, such as "executes," "is responsible for" or "has decision making power." An information item represents data, such as "output data." Finally, data can be sent or received by a function. Definition 1 shows a definition of the used variant of eEPCs based on the formalization of [Kin06]. It is important to note that the presented variant of eEPCs is a non-hierarchical EPC.

Definition 1 (Extended Event-driven Process Chain (eEPC))
An extended Event-driven Process Chain (eEPC) consists of events E, functions F, connectors C, control flow arcs $A \subseteq (E \cup F \cup C) \times (E \cup F \cup C)$, a set of names N, a labeling function l, organizational units O, information items I, an involved-with relation $i \subseteq F \times O$, a send relation $s \subseteq F \times I$ and a receive relation $r \subseteq F \times I$. The sets E, F, C, A, N, O and I are pair-wise disjoint.

The labeling function l assigns a name to an element of an EPC: $l : E \cup F \cup C \cup O \cup I \rightarrow N \cup \{and, or, xor\}$. A connector $c \in C$ may only have and, or, or xor assigned. Events, functions, organizational units and information items may only have an element out of N assigned.

$$M = (E, F, C, A, N, l, O, I, i, s, r)$$

To illustrate our mapping, a simplified online shop process is used. A customer places an order at an online shop, which processes the order. In parallel to the goods issue, the invoice is created and delivered. As soon as a payment of the customer is received, the payment is processed and the goods are delivered. Figure 2 presents the eEPC for the online shop.

3 The Fundamental Modeling Concept

The Fundamental Modeling Concept (FMC, [KW03, Tab05]) is a graphical modeling notation, which can be used to describe software systems on both the business and the technological level. FMC allows to visualize complex software systems as a system

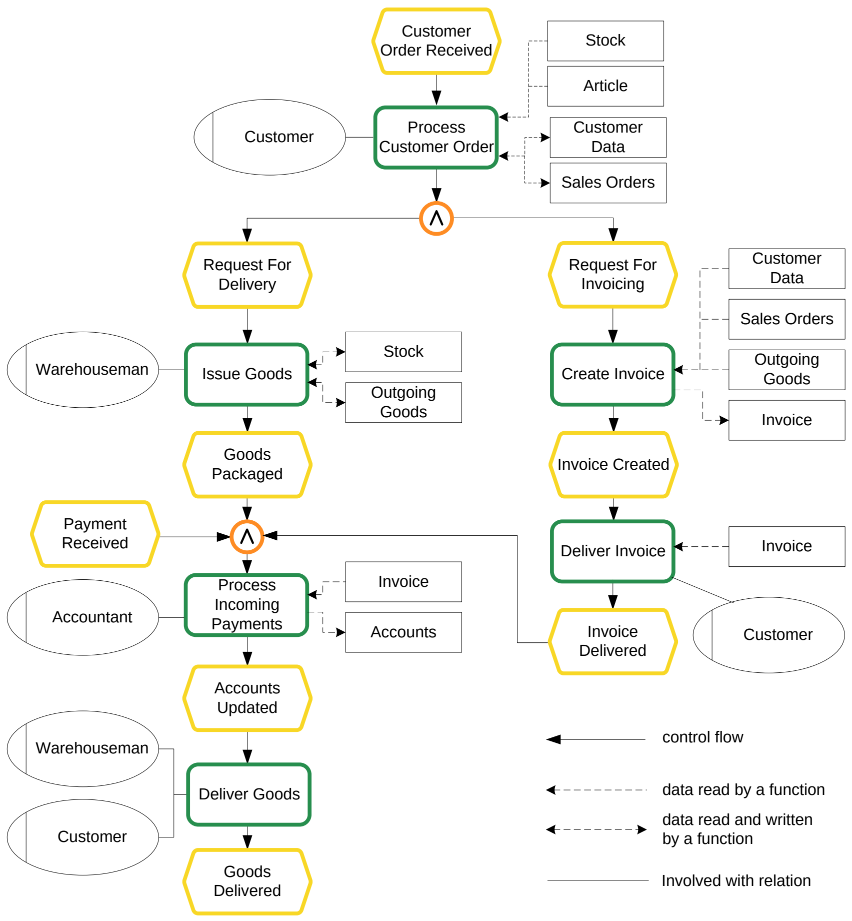


Figure 2: EPC for the example online shop

landscape which provides an overview on the used software, hardware and their relation to each other. FMC was introduced as an intuitive architectural description language to document and communicate a system architecture. It is designed to be simple and universal. Unlike UML [Obj07], FMC describes the structure of a system in a way that is independent of any implementation paradigm and that is on a higher level [KW03].

FMC distinguishes between three structural types: the data diagram, the process diagram and the block diagram. The data diagram is used to capture the relational data model and is similar to entity relationship diagrams. The process diagram expresses process structures using Petri nets. The block diagram is used to capture the components and their relationships. Since both FMC data diagrams and Petri nets are not used by business users, we focus on the block diagrams.

3.1 FMC Block Diagrams

FMC block diagrams are also called landscape architectures and provide an overview of the composite structure of systems [AK07]. In the following, we present all details of FMC block diagrams and present a formalization. The formalization itself does not include the rendering of FMC diagrams and excludes details which are not necessarily needed in the transformation.

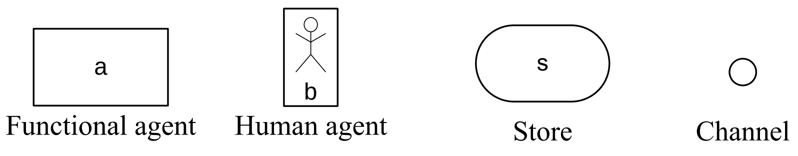
FMC block diagrams consist of components and their relationships. A component can be either active or passive. AC denotes the set of active components. An active component is capable of performing functionality. Active components are divided into functional agents and human agents. Functional agents represent functions with a defined mapping between input and output data. Human agents represent human actors. The function $\mathsf{t}_{\mathsf{AC}} : \mathsf{AC} \to \{\mathit{functional}, \mathit{human}\}$ returns the type of an active component. The function $\mathsf{n}_{\mathsf{AC}} : \mathsf{AC} \to \mathcal{N}$ returns the name of an active component, where $\mathcal{N}$ denotes the set of all names. An active component reads, writes or both reads and writes data.

PC denotes the set of passive components. A passive component is a storage or a channel. Note that passive components are also called “locations” to underline that data is located there. A storage is used by active components to store data and a channel is used for communication purposes between at least two active system components. Storages store data durably. A storage may contain triggers which trigger readers of the storage when a data item has been changed. Triggers do not have a visual representation and are therefore excluded from the formalization.

In contrast to storages, channels do not store data durably. Thus, channels can be regarded as telephone wires: The data is available at one point in time and will be lost afterwards. Hence, it is important that some active component reads it while it is there. The function $\mathsf{t}_{\mathsf{PC}} : \mathsf{PC} \to \{\mathit{storage}, \mathit{channel}\}$ returns the type of a passive component. Thus the implicit sets of storages PC_{S} and channels PC_{C} are defined:

$$\mathsf{PC}_{\mathsf{S}} = \{s \mid s \in \mathsf{PC},\ \mathsf{t}_{\mathsf{PC}}(s) = \mathit{storage}\}$$
$$\mathsf{PC}_{\mathsf{C}} = \{c \mid c \in \mathsf{PC},\ \mathsf{t}_{\mathsf{PC}}(s) = \mathit{channel}\}$$

The function $\mathsf{n}_{\mathsf{PC}} : \mathsf{PC} \to \mathcal{N} \cup \{\epsilon\}$ returns the name of a passive component. Storages must take a name and channels may be unnamed (denoted by ϵ). Note that the set $\mathcal{N}$ may not contain ϵ. The graphical notation of FMC components is presented in figure 3.



(a) Active Components (b) Passive Components

Figure 3: FMC Components

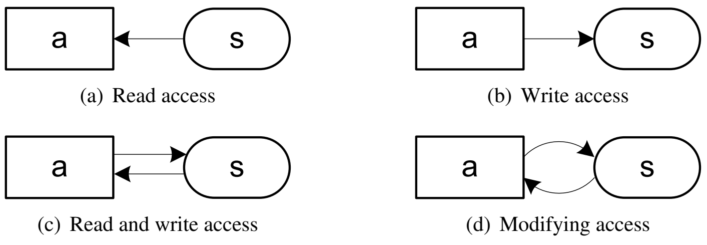
Active and passive components may be connected via connectors denoting their relationship. A relationship is read, write and modify. Read and write are both used at channels and storages. “Modify” may be used at storages to denote that an item in the storage is modified. FMC allows to model channels used for request-response. Therefore “rri” and “rrp” are introduced for channels. The initiator is connected with “rri” to the channel and the replying agent with “rrp”. Note that it is allowed to connect an agent to storage with both read and write relations. This states that the agent reads data and writes possibly different data. The set R contains different kinds of relationships: $\mathsf{R} = \{\mathsf{r}, \mathsf{w}, \mathsf{mod}, \mathsf{rri}, \mathsf{rrp}\}$. The set E is the set of all connectors.

$$\begin{aligned}\mathsf{E} \subseteq\, & \mathsf{AC} \times \{\mathsf{r}, \mathsf{w}, \mathsf{mod}\} \times \mathsf{PC_S} \cup \\ & \mathsf{AC} \times \{\mathsf{r}, \mathsf{w}, \mathsf{rri}, \mathsf{rrp}\} \times \mathsf{PC_C}\end{aligned}$$

Since a channel can be either used as request-response channel, as unidirectional channel or bidirectional channel, following property has to be satisfied by E:

$$\begin{aligned}&\forall (a, r, c) \in (\mathsf{E} \cap \mathsf{AC} \times \mathsf{R} \times \mathsf{PC_C}) : \\ &r \in \{\mathsf{r}, \mathsf{w}\} \;\Rightarrow \nexists\, (a, r', c) \in \mathsf{E} : r \in \{\mathsf{rri}, \mathsf{rrp}\} \wedge \\ &r \in \{\mathsf{rri}, \mathsf{rrp}\} \;\Rightarrow \nexists\, (a, r', c) \in \mathsf{E} : r \in \{\mathsf{r}, \mathsf{w}\} \wedge \\ &r = \mathsf{rri} \;\Rightarrow \nexists\, (a, r', c) \in \mathsf{E} : r = \mathsf{rrp} \wedge \\ &r = \mathsf{rrp} \;\Rightarrow \nexists\, (a, r', c) \in \mathsf{E} : r = \mathsf{rri}\end{aligned}$$

Figure 4 presents the graphical representation of connectors between active components and storages and Figure 5 presents the graphical representation of connectors between active components and channels.



(a) Read access (b) Write access

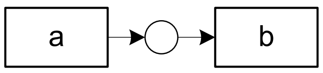
(c) Read and write access (d) Modifying access

Figure 4: Relationships between active components and storages

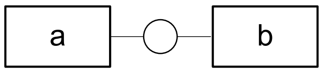
In short, a FMC block diagram is bipartite graph G, where the sets AC and PC form the nodes and E forms the edges between the nodes.

$$G = (\mathsf{AC}, \mathsf{PC}, \mathsf{E}, \mathsf{RR}, \mathcal{N}, \mathsf{t_{AC}}, \mathsf{t_{PC}}, \mathsf{n_{AC}}, \mathsf{n_{PC}})$$

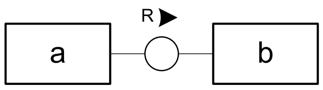
.



(a) Unidirectional channel. Active component a has write access to the channel, whereas active component b has read access to the channel



(b) Bidirectional channel. Both active components a and b have read and write access to the channel



(c) Request-response channel. a is the initiator and b replies to the request.

Figure 5: Relations between active components and channels

3.2 Business Landscapes

FMC can be used to model the architecture from a business and from an IT point of view on any granularity level. Therefore, it is important to state what aspects should be captured by the block diagram. Our work focuses on the business aspect. So called “business landscapes” are used in the requirements engineering and are usually designed by business people [AK07]. Business landscapes do not depict technical information such as servers, clients or databases. The purpose of business landscapes is to visualize

- business functionalities,
- business data,
- business roles and
- their relationships with each other.

Human agents represent business roles, such as members of the sales department. *Functional agents* represent business functionality. *Storages* can either represent stored master data or dynamic data. *Channels* connect agents with each other. The semantics of a connection established via a channel is that information is flowing between the two connected agents. If a storage is connected with two agents, it is not necessary to additionally connect these two agents with a channel.

Figure 6 presents an example business map. It shows the business perspective on the architecture of an online shop.

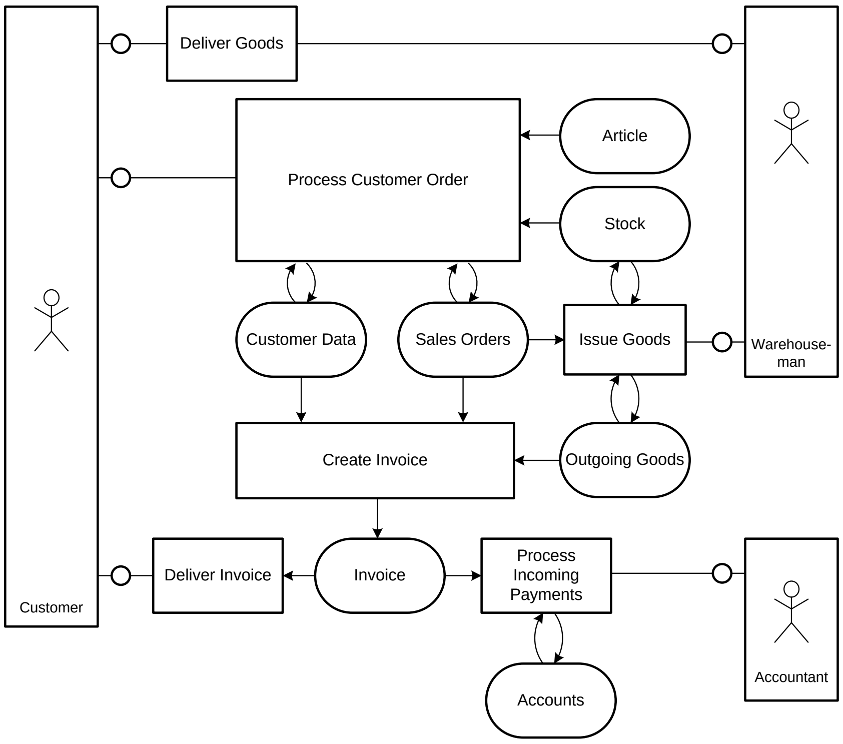


Figure 6: FMC business landscape of an online shop, adopted from [AK07]

3.3 Pure IT Landscapes

Pure IT landscapes are FMC block diagrams capturing technological aspects. They show software components and hardware components and their relationships. For example, a software component is a database and a hardware component is a Unix server. Pure IT landscapes can be used to define new systems and to document already existing ones. New systems can be designed by using architectural patterns, depending on what the requirements on the system are. If existing systems are documented, the resulting landscape can be used to identify bottlenecks and to determine changes needed for improvement of the system. Figure 7 presents the pure IT landscape of the example online shop.

3.4 Relating Pure IT Landscapes with Business Landscapes

Business functionality and business data need an implementation. Therefore, business and pure IT landscapes need to be related. These relations are shown in a mixed landscape, called “IT landscape” [AK07]. IT landscapes are pure IT landscapes, where business functions and data are embodied in the technical components. This approach enables

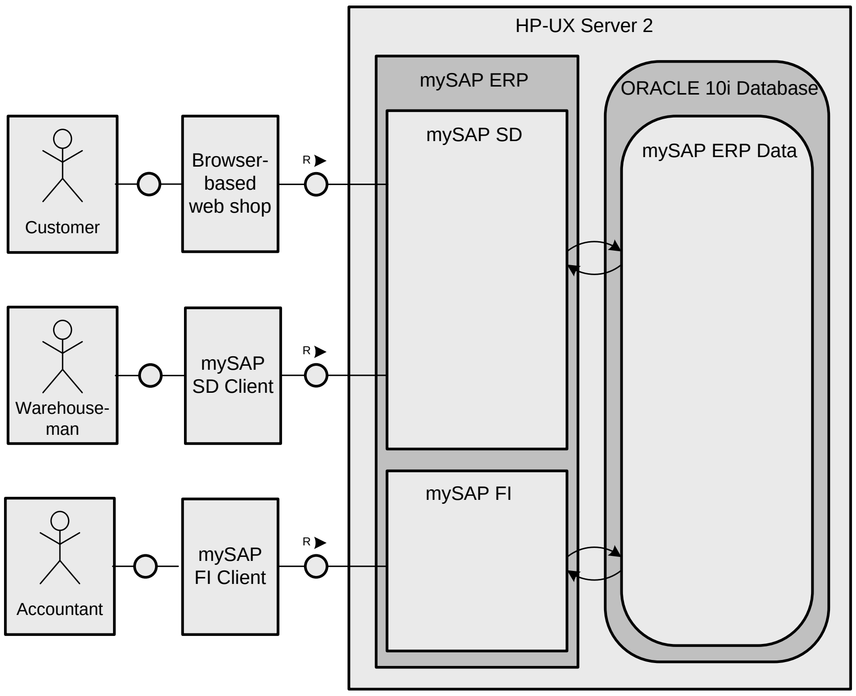


Figure 7: FMC pure IT landscape of an online shop

business analysts and IT professionals to start modeling independently of each other. Figure 8 presents the combined IT and business landscapes. In our application scenario, the IT landscape is the place, where the artifacts of the generated business landscape are placed in to uncover deficiencies of the existing pure IT landscape.

4 Mapping eEPCs to an FMC Block Diagram

Having defined the notation of eEPCs and FMC, we present how to map information from eEPCs to an FMC block diagram. We explain each step, show an algorithm and apply each step to the example online shop to illustrate the mapping. The algorithm maps an eEPC $M = (E, F, C, A, N, l, O, I, i, s, r)$ to a FMC block diagram $G = (\mathsf{AC}, \mathsf{PC}, \mathsf{ERR}, \mathcal{N}, \mathsf{t}_{\mathsf{AC}}, \mathsf{n}_{\mathsf{AC}}, \mathsf{n}_{\mathsf{PC}})$. We assume, that all sets in G are set to empty sets and all functions are undefined.

In FMC, organizational units are seen as business roles, which are represented in the FMC as human actors. The aim of the modeled EPC is to provide a business view on an executable workflow. This implies that each function of the EPC should be executed by a machine. Since a machine is represented as functional agent in FMC, we map functions to functional agents. An organizational unit is connected with a function by the relation “is

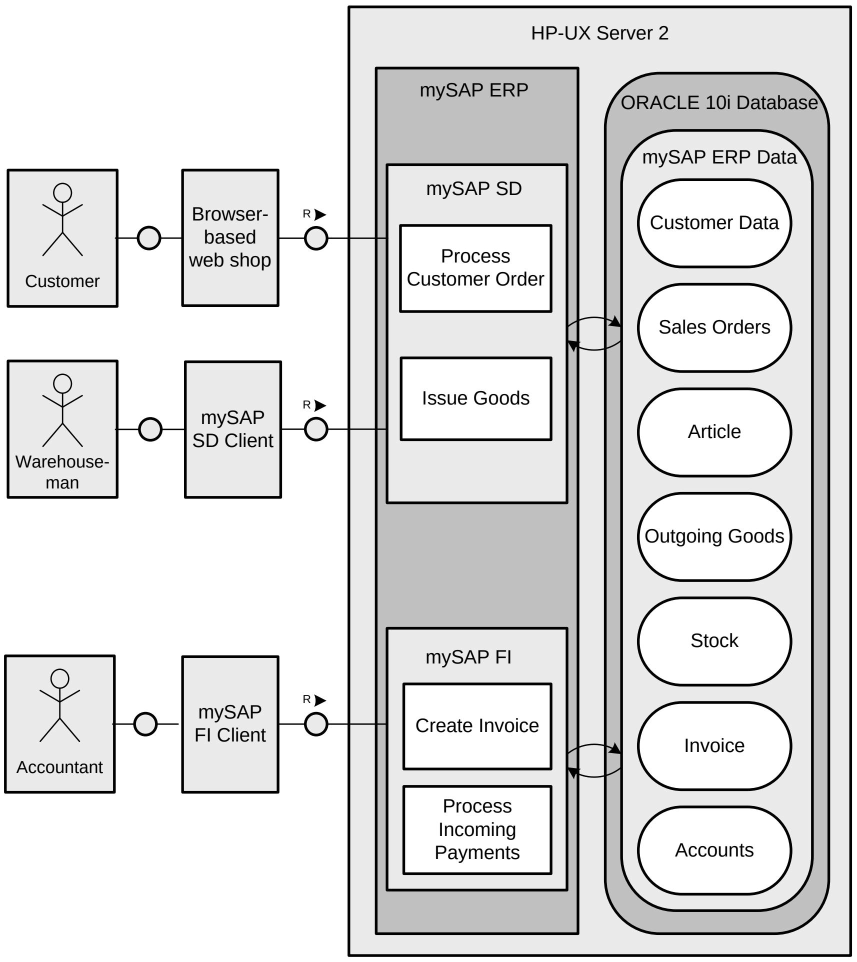


Figure 8: FMC IT landscape of an online shop [AK07]

involved with". That relation does not state any direction of the communication. Therefore, the relation between the human agents and functional agents is realized with a bidirectional channel. A bidirectional channel denotes that the agents are communicating with each other. The mapping of a function to a functional agent can also be interpreted, that the functional agent in the FMC block diagram is "supporting" to human agent to do his task. For example, a tool or a screen form support a human to perform his task.

It is important to note that the names of the artifacts are retained during the mapping. Figure 9 presents the mapping of functions, the involved organizational units and their relation to agents and channels connecting them. Algorithm 1 presents the algorithm. In the algorithm, we used the fact that a function $f : A \rightarrow B$ can also be represented as a set of tuples $f \subseteq A \times B$.

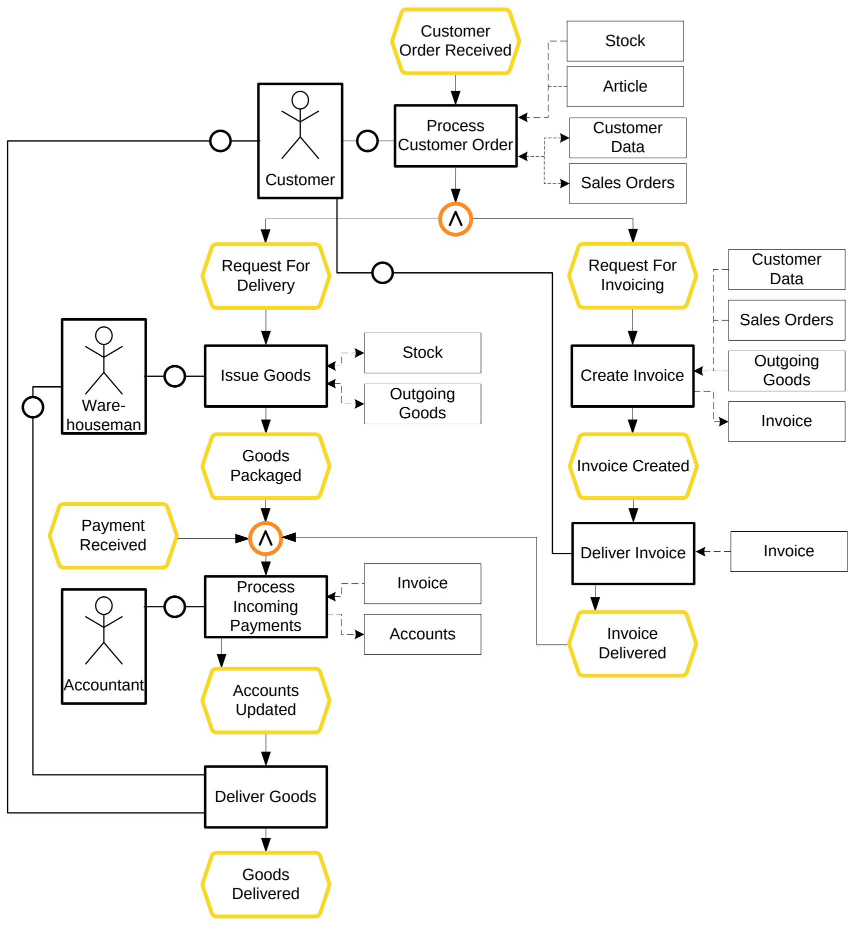


Figure 9: Functions mapped to functional agents. “is involved with” mapped to bi-directional channels.

An information item represents data. Since data is stored in storages in FMC, information items are mapped to storages. If a function receives an information item, it has to read the information item somewhere. Therefore, the read relation is mapped to a connection with the relationship “read”. By analogy, the write relation is mapped to a connection with the relationship “write”. If a functions both reads and writes an information item, both are summed up in the modifying access relationship. Figure 10 illustrates the mapping using the data accessed by the “process customer order” function as an example. The algorithm is presented in algorithm 2.

So far, the flow relation has not been mapped. Depending on the underlying IT infrastructure, explicit channels are needed to denote that a functional agent triggers a subsequent agent. Therefore, the connection between two functions (with no other function in between) can be mapped to a unidirectional channel. On the other hand, storages may contain triggers.

Algorithm 1 Mapping organizational units, functions and their relation to FMC. Input is an eEPC M, the output is a FMC block diagram G.

```
procedure CreateActors($M$,$G$)
    $\mathsf{AC} \leftarrow F \cup O$
    $\mathsf{t_{AC}} \leftarrow \{(ac, functional) \mid ac \in F\} \cup \{(ac, human) \mid ac \in O\}$
    $\mathcal{N} \leftarrow \{n \mid n \in l(e),\ e \in F \cup O\}$
    $\mathsf{n_{AC}} \leftarrow \{(ac, l(ac)) \mid ac \in \mathsf{AC}\}$
    for all $(f, o) \in i$ do
        Create a new channel $c$
        $\mathsf{PC} \leftarrow \mathsf{PC} \cup c$
        $\mathsf{t_{PC}} \leftarrow \mathsf{t_{PC}} \cup (c, channel)$
        $\mathsf{n_{PC}} \leftarrow \mathsf{n_{PC}} \cup (c, \epsilon)$
        $\mathsf{E} \leftarrow \mathsf{E} \cup (f, \mathsf{r}, c) \cup (f, \mathsf{w}, c)$
        $\mathsf{E} \leftarrow \mathsf{E} \cup (o, \mathsf{r}, c) \cup (o, \mathsf{w}, c)$
    end for
end procedure
```

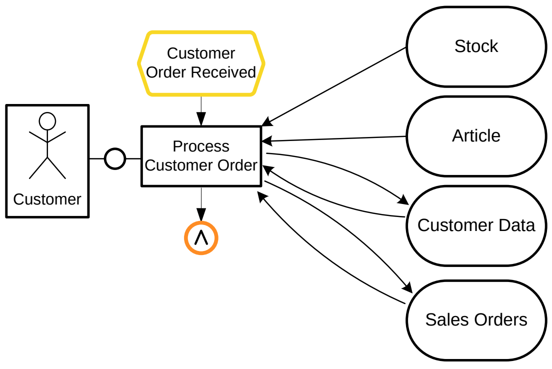


Figure 10: Information items mapped to storages

Algorithm 2 Mapping organizational units, functions and their relation to FMC. Input is an eEPC M and an FMC block diagram G. The block diagram G is modified and thus also an output.

```
procedure CreateStorages($M$,$G$)
    $\mathsf{PC} \leftarrow \mathsf{PC} \cup I$
    $\mathsf{t_{PC}} \leftarrow \mathsf{t_{PC}} \cup \{(i, storage) \mid i \in I\}$
    $\mathcal{N} \leftarrow \mathcal{N} \cup \{n \mid n \in l(i),\ i \in I\}$
    $\mathsf{n_{PC}} \leftarrow \mathsf{n_{PC}} \cup \{(i, l(i)) \mid i \in I\}$
    $RW \leftarrow s \cap r$                                // Function sending and receiving
    for all $(f, i) \in RW$ do
        $\mathsf{E} \leftarrow \mathsf{E} \cup (f, \mathsf{mod}, i)$
    end for
    $W \leftarrow s \setminus RW$                                   // Function only sending
    for all $(f, i) \in W$ do
        $\mathsf{E} \leftarrow \mathsf{E} \cup (f, \mathsf{w}, i)$
    end for
    $R \leftarrow r \setminus RW$                                 // Function only receiving
    for all $(f, i) \in R$ do
        $\mathsf{E} \leftarrow \mathsf{E} \cup (f, \mathsf{r}, i)$
    end for
end procedure
```

For example, an inserted information item can trigger the subsequent agent. Hence, an additional channel is unnecessary in this case. Since triggers are currently not included in the metamodel of FMC, our mapping is not aware of triggers. Furthermore, the Business Process Execution Language (BPEL) is used to drive the business process in a web services architecture [WCL+05]. Thus, the services (modeled by functional agents) do not have to be aware of other services in the business process. Thus, we do not map the connections between functions to channels. As a result, events between functions are not mapped either. There are two mapping options for events without a preceding or subsequent function: (i) induce a human actor and connect it with the actor representing the preceding/subsequent function or (ii) do not map it to a FMC representation. Option (i) is not consistent with the exclusion of channels between functional agents in the mapping. Therefore, option (ii) is taken, which leads to the FMC business landscape presented in figure 6.

Algorithm 3 presents the complete algorithm, which first initializes the target FMC block diagram and calls CreateActors and CreateStorages.

5 Related Work

There exists work on the transformation of graphically modeled business processes to executable workflows (e.g., [ODtHvdA07, MLZ06]). These works do not deal with architecture generation, but can be extended with our contribution. [Asc07] presents how a

Algorithm 3 Mapping an eEPC to a FMC block diagram. Input is an eEPC M, the output is a FMC block diagram G.

```
procedure MAP($M$,$G$)
    $\mathsf{AC}, \mathsf{PC}, \mathsf{RR}, \mathcal{N}, \mathsf{t}_{\mathsf{AC}}, \mathsf{n}_{\mathsf{AC}}, \mathsf{n}_{\mathsf{PC}} \leftarrow \emptyset$
    CREATEACTORS($M$, $G$)
    CREATESTORAGES($M$, $G$)
end procedure
```

FMC business landscape can be manually created out of an EPC. The generated business landscape contains more information than the EPC alone. For example, actors performing functionality in the same area, are nested in an actor representing functionality in that area. Our work extends that work by automating the mapping of the EPC elements to FMC business landscapes, which is a basis for the subsequent step of structuring and simplifying the business landscape.

[Mol05] describes a process-oriented application landscape with IT modeling as one aspect. The work states that an IT model cannot be automatically generated out of a business process due to different modeling domains. Instead of automatic generation, required control structure and data can be mapped to generic methods and objects. Our work supports this mapping by automatically generating required artifacts. These artifacts can then be placed in the existing IT landscape.

The Rational Unified Process [Kru04] is an engineering process, which describes phases for software development. Processes can be modeled as EPCs in the "business modeling discipline". In the following discipline "Requirements discipline", the generated business landscapes can be used as communication vehicle between the developers and the customers.

6 Conclusion and Outlook

We presented the concept of mapping extended Event-driven Process chains (eEPCs) to FMC business landscapes. This enables a top-down approach for modeling IT infrastructures: First, the eEPC for the business process is modeled. Afterward, the presented mapping is used to derive a FMC business landscape. This landscape can then be used as basis for creating an IT landscape supporting the modeled business process. The approach can also be used to identify the gaps between the existing IT infrastructure and the infrastructure needed for process execution.

Another application scenario is to map multiple eEPCs to a single business landscape. This business landscape captures the requirements of all given processes at one place. Since the generated business landscape provides another view on the modeled processes, the landscape can be used as additional validation of the modeled processes by the business process designers.

The generated business landscape is consistent with the business process defined in eEPC. That means, the required business functions, the required storages and the participating human actors are – by construction – all contained in the business landscape. Thus, defining or checking the IT using the generated business landscape ensures that all required business functionality is covered by the IT.

Ongoing work is the implementation of the approach in Arcway Cockpit [AG07] and a customer evaluation. Future work is the support of eEPCs containing information about the IT infrastructure. For example, a machine can execute a function, which results in a nesting of the generated functional agents. We plan to define FMC4SOA, which includes special markers for an SOA IT infrastructure at FMC block diagrams. This allows for using EPCs with technical details (e.g., [ZM05]) can be used as input of our approach. The technical details are then stored in an FMC4SOA landscape.

Acknowledgments

Oliver Kopp is funded by the German Federal Ministry of Education and Research (project Tools4BPEL, project number 01ISE08).

References

[AG07] Arcway AG. Homepage of Arcway Cockpit, 2007. `http://www.arcway.com`, visited 2007-09-15.

[AK07] P. Aschenbrenner and F. Keller. Visual Requirements Engineering for IT-Projects, 2007. `http://www.arcway.com/fileadmin/arcwaydateien/pdf/ARCWAY_VRE_Whitepaper.pdf`, visited 2007-09-15.

[Asc07] P. Aschenbrenner. Business Process driven Requirements Engineering, 2007. `http://www.arcway.com/fileadmin/arcwaydateien/pdf/BPRE_Whitepaper.pdf`, visited 2007-09-15.

[Kin06] E. Kindler. On the semantics of EPCs: Resolving the vicious circle. *Data Knowl. Eng*, 56(1):23–40, 2006.

[KNS92] G. Keller, N. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Technical Report Heft 89, Universität des Saarlandes, 1992. Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi).

[Kru04] P. Kruchten. *The Rational Unified Process*. Addison-Wesley, third edition, 2004.

[KW03] F. Keller and S. Wendt. FMC: An Approach Towards Architecture-Centric System Development. In *ECBS 2003*, pages 173–182. IEEE Computer Society, 2003.

[MLZ06] J. Mendling, K.B. Lassen, and U. Zdun. Transformation Strategies between between Block-Oriented and Graph-Oriented Process Modeling Languages. In F. Lehner, H. Nösekabel, and P. Kleinschmidt, editors, *Multikonferenz Wirtschaftsinformatik 2006 (MKWI 2006)*, volume 2, pages 297–312. GITO-Verlag Berlin, 2006. XML4BPM Track.

[Mol05] M. Molter. Die prozessorientierte Applikationslandschaft. In A.-W. Scheer, W. Jost, and K. Wagner, editors, *Von Prozessmodellen zu lauffähigen Anwendungen*, pages 151–172. Springer, 2005.

[Obj07] Object Management Group. *UML 2.1.1 Superstructure Specification*, February 2007.

[ODtHvdA07] C. Ouyang, M. Dumas, A.H.M. ter Hofstede, and Wil M.P. van der Aalst. Pattern-based translation of BPMN process models to BPEL web services. *International Journal of Web Services Research (JWSR)*, 2007.

[STA05] A.-W. Scheer, O. Thomas, and O. Adam. Process Modeling using Event-Driven Process Chains. In M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede, editors, *Process-Aware Information Systems*, pages 119–146. Wiley & Sons, 2005.

[Tab05] P. Tabeling. *Softwaresysteme und ihre Modellierung. Grundlagen, Methoden und Techniken*. Springer, Berlin, 2005.

[WCL$^+$05] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D.F. Ferguson. *Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall PTR, 2005.

[ZM05] J. Ziemann and J. Mendling. EPC-Based Modelling of BPEL Processes: a Pragmatic Transformation Approach. In *Proceedings of the 7th International Conference Modern Information Technology in the Innovation Processes of the Industrial Enterprises (MITIP 2005)*, Genova, Italy, September 2005.