

# **EMC – A MODELING METHOD FOR DEVELOPING WEB-BASED APPLICATIONS**

Peter Rittgen

Institute of Business Informatics, University Koblenz-Landau, Rheinau 1, D-56075 Koblenz

email: rittgen@uni-koblenz.de, phone: +49-261-287-2535, fax: +49-261-287-2521

## **ABSTRACT**

Early information systems were mainly built around secondary, administrative processes of the value chain (e.g. accounting). But since the internet came into use, more and more primary processes have become accessible to automation: customer acquisition, ordering, billing and, in the case of intangible goods such as software, even delivery. Hence an increasing part of an enterprise has to be modeled and a substantial part thereof is implemented, usually in an object-oriented programming language like Java. To facilitate this complex task, the MEMO methodology (Multi-perspective Enterprise MOdeling) allows the description of the enterprise on three levels – strategy, organization and information system – and from four angles – process, structure, resources and goals. All partial models for the views are integrated via a common object-oriented core. In this framework we suggest a modeling method for the IS layer, the Event-driven Method Chain (EMC). It is based on the Event-driven Process Chain (EPC) by Scheer, which we adapt to fit both the MEMO methodology and the object-oriented paradigm thus making it suitable for the development of web-based applications. To illustrate this we use the example of a software trading company.

## **1. INTRODUCTION**

Early information systems were mainly built around secondary, administrative processes of the value chain (e.g. accounting). But since the internet came into use, more and more primary processes have become accessible to automation: customer acquisition, ordering, billing and, in the case of intangible goods such as software, even delivery. Hence an increasing part of an enterprise has to be modeled and a substantial part thereof is implemented. To create such an information system and to adapt it constantly to a changing environment requires a much more efficient software development process than the one suggested by the traditional methods, namely the separation into the phases analysis, design and implementation where each phase is usually performed by a different team, each relying on the documents produced in the previous phase (possibly with backtracking). In these approaches, the coupling between the phases is weak: changes to an analysis model typically require a substantial reorganisation of the design models, which in turn slows down software development considerably. The ARchitecture of Integrated information Systems (ARIS, [Scheer 99]) is one such traditional method with a focus on analysis. It received substantial attention both from researchers and practitioners thanks to its close relation to the SAP suite of business applications.

Now, there are several reasons why such methods are not suitable for the development of web-based applications (leading to corresponding requirements):

1. The increasing number of automated business processes requires the integration of these processes with the relevant data. But the parts (views) of conventional models are only loosely coupled (e.g. the data and process models of ARIS). A suitable method for developing web-based applications should integrate partial models, especially the process and the data/object model (*model integration*).
2. Existing methods do not cater for the needs of object orientation. But the predominant use of object-oriented programming languages in developing web-based software demands the compatibility of the modeling method with *object-oriented* concepts.
3. Traditional software development is usually quite slow. But electronic markets require a quick adaptation of web applications to changing needs.
4. The development of design models typically consists of reinventing the analysis models in a different (more formal) language. A smooth transition from analysis to design, by merely refining the existing analysis models, is preferable (*phase integration*).

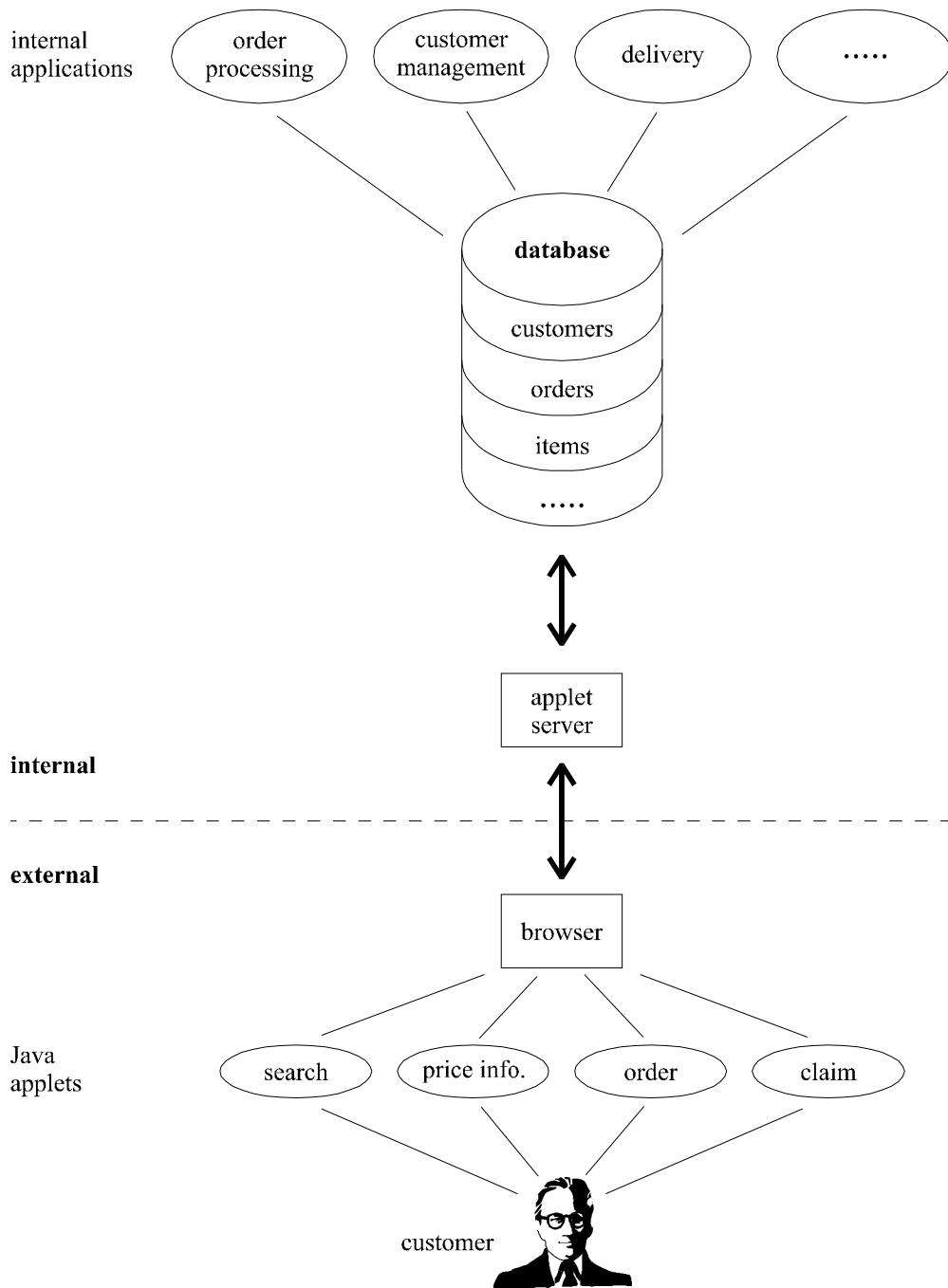
To solve these problems, we make use of the MEMO framework, which represents a multi-perspective approach to enterprise modeling where all views are strongly connected via a common object-oriented core. Within this framework, the integration of processes and their relevant data is achieved by an analytical, object-oriented process definition language called Event-driven Method Chain (EMC). Apart from model integration, EMCs also facilitate the transition from analysis to design by providing a suitable level of abstraction/formalization, hence speeding up the software development process.

## 2. AN EXAMPLE SETTING

To illustrate the problem of developing a web-based application, we consider the example of a young mail-order company trading software products. Up to now, they were organized in a more or less conventional way: customers ordered via phone, fax or surface mail. Orders were processed manually and then entered into a database. The products were stocked in physical form as CD-ROMs. Stock management was done with the help of a stand-alone system. Delivery was effected by conventional posting, payment by cheque, credit card or money order.

Now, this company plans to operate over the internet. Apart from offering new services (such as ordering via the world wide web and downloading of the ordered product), this also requires substantial reorganization: e.g. the isolated information systems for ordering and stocking have to be integrated to allow the potential customer a combined search for price and availability of a product. The head of IT is therefore asked to draw up a sketch of the principal architecture of the new system (see fig. 1). It consists of a central database containing information about customers, orders, items and so on. All applications, internal and external, operate on this database. Internal applications are the ones used only by the staff of the company, such as order and customer management, delivery etc. The external applications can also be accessed by the (potential) customers. They are made accessible to the world by an applet server feeding the user's browser.

The next sections will show how such an information system can be analyzed and designed rapidly with the help of the EMC method. This method integrates the different views on a system as well as their migration from analysis to design models because it adheres to the multi-perspective methodology for enterprise modeling (MEMO) as outlined in the following section.



*Fig. 1: Example architecture of a web application for a software trading company*

### 3. MULTI-PERSPECTIVE ENTERPRISE MODELING (MEMO)

In the early phase of analysis, the modeler of any application is typically confronted with a yet largely unstructured problem domain. This applies to web-based applications in particular, which involve a complete reorganization of a substantial part of the company: processes to be performed over the web have to be designed newly, related internal processes have to be adapted accordingly. These changes affect not only the information system itself but also the organization as a whole resulting in a significant strategical impact of the corresponding decisions. Hence a modeling methodology should distinguish three levels of an enterprise: strategy, organization and information system (see fig. 2).

But even if we restrict attention to one of the levels, too much complexity remains to be handled by one model only. Therefore each level is further subdivided into the following four foci:

- structure: static description of the domain objects and their relations
- process: dynamic representation of the system
- resources: means required to execute a process
- goals: results to be achieved and their relations

The resulting 12 partial models span the 4×3 matrix of the views of MEMO, the acronym of Multi-perspective Enterprise MOdeling [Frank 97]. Fig. 2 gives iconized examples for the 12 views. For example, the strategical process model (SPM) consists of a value chain á la Porter [Porter 85] where the primary processes (procurement, production, distribution, marketing) form the basic chain (arrow-shaped boxes) with the subsidiary (supporting) activities attached to them (the flags). On the organizational and IS levels, an EPC-like language for modeling process is used. The OSM is represented in the form of an organizational chart and the ISM is an object class model. The remaining views in the matrix have not been covered yet.

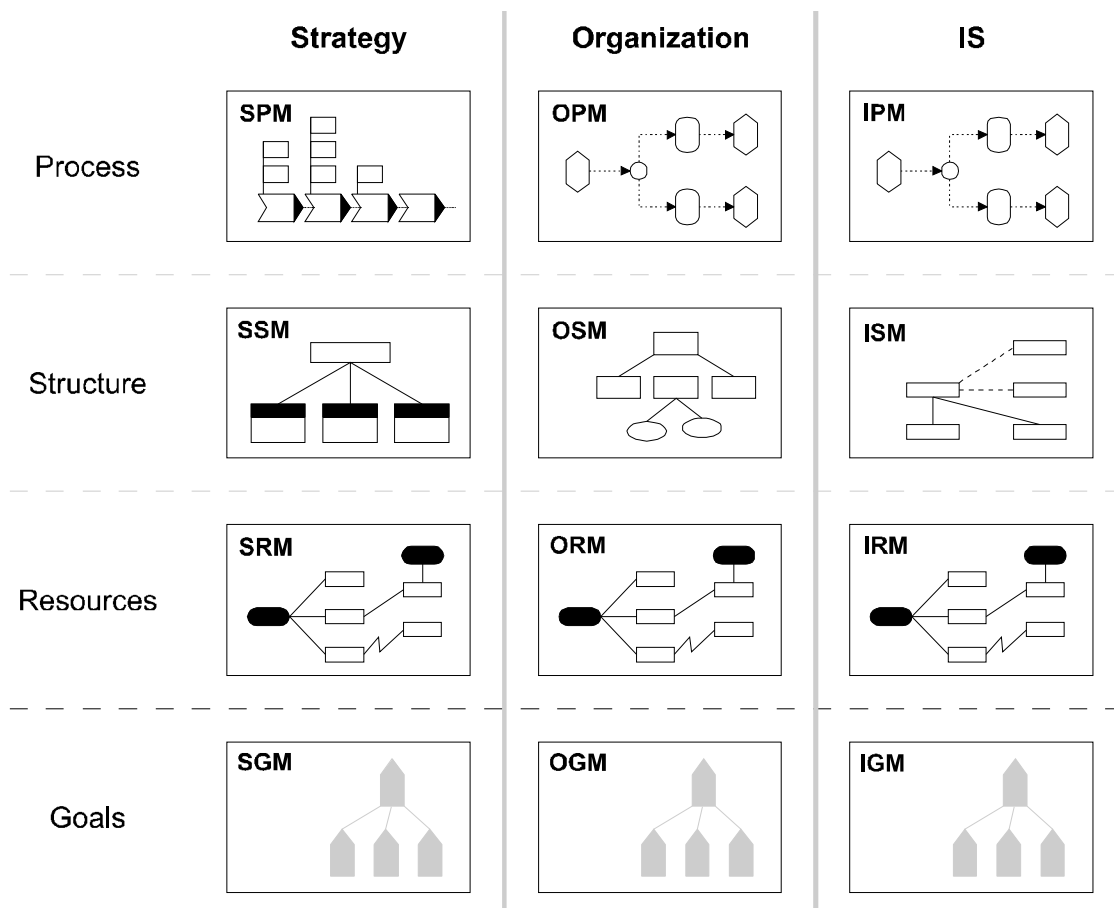


Fig. 2: Perspectives on an enterprise (MEMO)

Here we focus on developing an integrated modeling language for the IS level (abstracting from the goal view for the time being). We call this language EMC (Event-driven Method Chain). It covers IPM (IS Process Model), IRM (IS Resource Model) and the integrative part of ISM (IS Structure Model). The details of the structure are thereupon specified with the help

of MEMO-OML (MEMO Object Modeling Language, see [Frank 98]). Here MEMO-OML is explained only insofar as it is necessary to understand the integration of all three views.

We start from the assumption that the processes of a problem domain are rather easier to identify than the more abstract objects. Hence we put the process focus in the center of the EMC language and suggest that an initial EMC is drawn before the corresponding object model. In fact, the EMC even helps in the construction of the object model. The basic dynamic element of an EMC is the method (or service) which is linked to the objects involved in providing this service. The objects themselves and their relations are defined in MEMO-OML. We establish the resource focus by attaching the resources to the method which requires them. Fig. 3 shows how the individual foci are represented in the models according to the MEMO methodology (i.e. a study of methods).

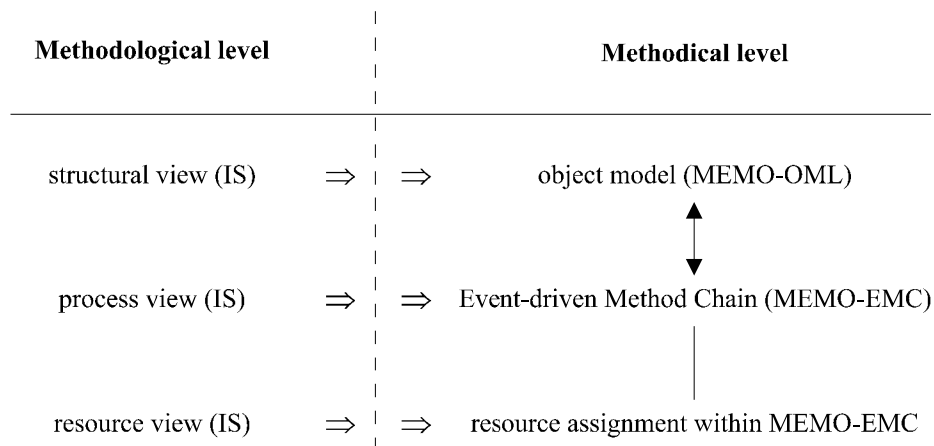


Fig. 3: Representing the MEMO views in the models

The process modeling language EMC is based on the Event-driven Process Chains (EPCs) of ARIS. Although this language exhibits major shortcomings as demonstrated in the next section, we do not reject it outright because it has proved its ease of use in practice: it is the preferred choice of consultants and large IT departments, and it is a must for companies introducing SAP. Hence a process language based on EPCs has a better chance of being accepted by the practitioner than some completely new artefact. Still, the shortcomings have to be overcome to make EPCs fit into the MEMO framework (thus achieving model integration), to align them to the object-oriented paradigm and to ensure an unambiguous interpretation of the analysis models by the designers thus achieving phase integration (see points 1-4 in the introduction). The result is the EMC language described in section 5.

#### 4. SHORTCOMINGS OF EPCS

When modeling business processes in ARIS, we identify core processes of the company and represent them as EPCs. An EPC consists of a strictly alternating sequence of events (“invoice arrived”) and functions (or processes) such as “enter invoice” (hence its name). In addition, alternative or concurrent processes can be specified with the help of connectors (XOR, OR, AND). The model either captures existing processes or specifies planned ones, and it can be employed to reengineer inefficient business processes.

The syntax of a standard EPC is given quite accurately by [KT 97]. It is generally accepted by the majority of the literature on EPCs. Its 14 rules include:

- K1: There are no isolated nodes.
- K3/4: Functions and events have exactly one incoming and one outgoing edge (except start and end events).
- K6: Connectors are either splits (1 input, several outputs) or joins (several inputs, 1 output).
- K8/9: An event is always followed by a function and vice versa (modulo connectors).

Concerning semantics, there is little unanimity. It is given only roughly (in a verbal form) in the original publication by Scheer [Scheer 92]. Later there have also been attempts to give EPCs a formal semantics, e.g. [CS 94], [Rump 97] and [LSW 97]. But all approaches differ considerably, i.e. they attribute different meanings to the same EPC in many cases. Many of these discrepancies stem from diverging interpretations of the logical connectors, in particular of the (X)OR join. So assuming the join has input paths *a* and *b* (like in fig. 4, on the left) the following ambiguities may arise:

- Does the (closing) join have a matching (opening) split? If so, the semantics is usually taken to be “wait for the completion of all paths activated by the corresponding split”. But how can we identify a matching split?
- If there is no matching split, there are three symmetrical interpretations (i.e. interpretations not distinguishing between *a* and *b*) of an OR join (see fig. 4, on the left):
  1. Wait for the completion of all **activated** paths (called wait-for-all).
  2. Wait only for the path that is completed first and ignore the second (called first-come).
  3. Trigger the outgoing path *c* on each completion of *a* or *b* (called every-time).
- In the case of no matching split, there is also a problem with the XOR join: should it block if both *a* and *b* have been activated or should it operate in every-time mode?

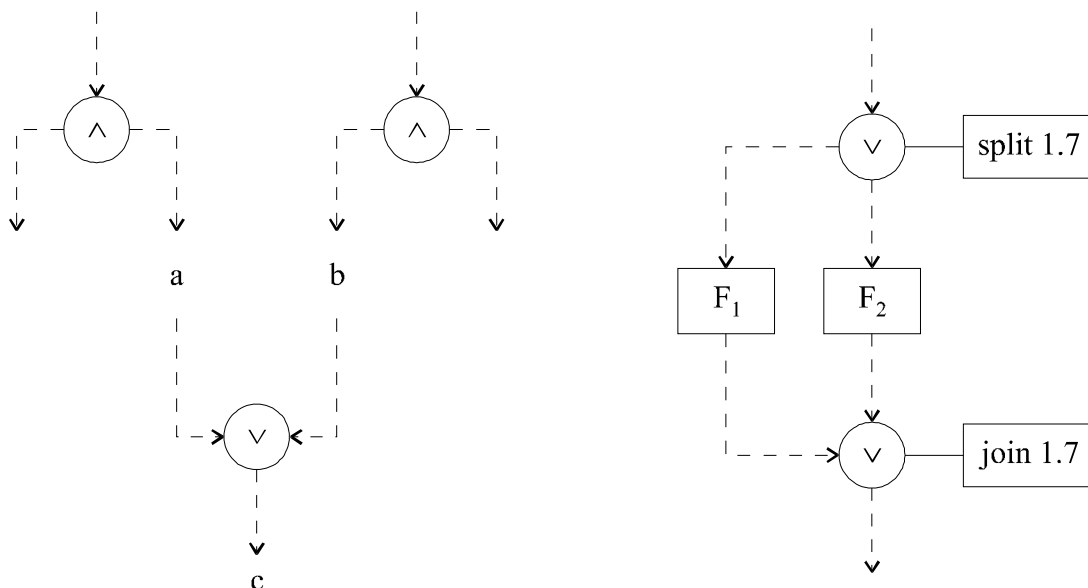
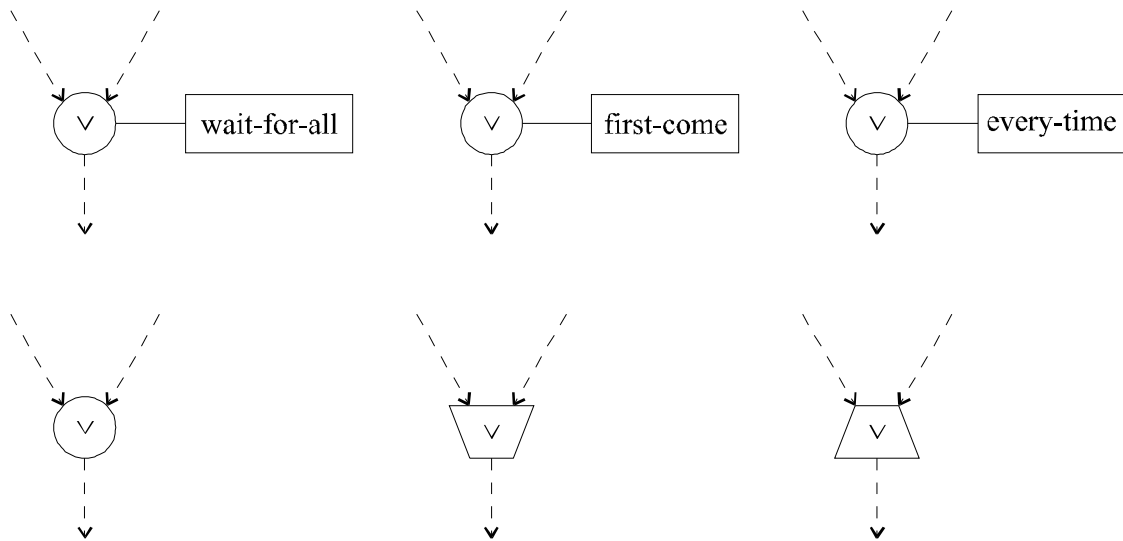


Fig. 4: Ambiguous OR join (on the left), matching split and join (on the right)

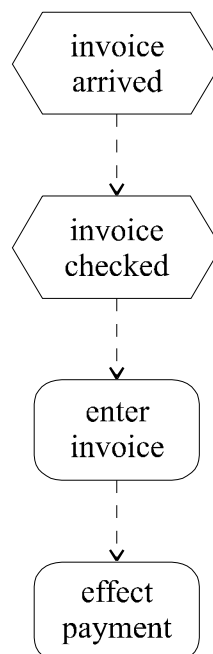
The semantical shortcomings mentioned above can be remedied by extending the syntax of the connectors. We suggest allowing the modeler to add a comment flag to a connector. This flag can uniquely identify matching connectors (see fig. 4, on the right), and it may serve to

clarify the intended meaning of an unmatched join (see fig. 5). Alternatively, the meaning can also be encoded in the connector symbol itself: a standard OR symbol to denote wait-for-all, a funnel-like trapezoid for first-come and an inverse trapezoid for every-time.



*Fig. 5: Making the OR join unambiguous*

Beyond their semantical shortcomings, EPCs also exhibit some deficiencies originating in the syntactical domain. Empirical studies like [Speck 98] have shown that particularly middle and upper management people consider the strict alternation between events and functions as too restrictive. They find it hard to identify the necessary events on an abstract level of process description.



*Fig. 6: A non-alternating sequence of events and functions*

We suggest dropping this syntactical requirement as dummy events might always be added later if this proves to be necessary. On a conceptual level, there are good reasons to be able to omit events as fig. 6 shows: if “enter invoice” and “effect payment” are performed as one unit (i.e. uninterruptedly) there is no need for an event to trigger the second activity. A similar

reasoning leads us to allow two (ore more) consecutive events (see also fig. 6). If we change the syntax of EPCs as outlined in the last two paragraphs, we arrive at the so-called modified EPCs (or modEPCs).

## 5. THE EVENT-DRIVEN METHOD CHAIN

After having effected all the modifications suggested in the previous section, the resulting modEPCs provide an excellent basis for the process model of a web application. A unique and formal interpretation can now be given for any EPC, e.g. in the form of a Petri net (see [Ritt 99]). Hence modEPCs can be understood unambiguously by the protagonists of the following design phase. This leads to less mistakes in the design model and less backtracking from design to analysis thus fulfilling requirements 3 (faster software development) and 4 (phase integration) of section 1.

If we put together modEPCs (process focus), classes with their attributes (structural focus) and the resources (resource focus), we arrive at the Event-driven Method Chain (EMC). Its syntax is shown in fig. 7.

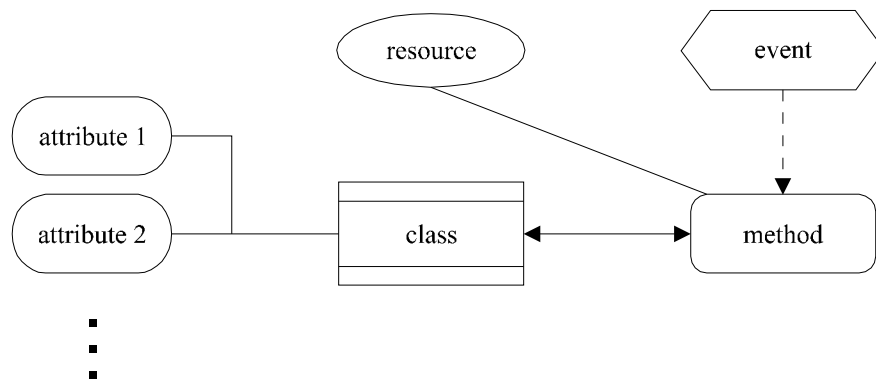


Fig. 7: General syntax of an EMC

The function of a modEPC is now performed by some object and hence called the *service* (or method) provided by this object. Apart from this, the process part of an EMC, consisting of events, methods, connectors and control flow arcs (dashed arrows), follows exactly the syntax of modEPCs (section 4). The object classes involved in providing the service are linked via a double-headed, solid arrow. A class can have attached to it a list of attributes. Classes are also called the internal resources required by a service because they form an integral part of the information system. An external resource is denoted by an oval. It is connected via a solid line to the method requiring it.

Now, classifying objects into classes sharing common attributes and methods is an important object-oriented feature. Together with the characteristics provided by the MEMO-OML (see [Frank 98]), our approach fulfils all requirements of object-orientation and hence point 2 of section 1. But how does EMC work in our example setting?

Fig. 8 shows a part of the EMC for the order processing within the web application outlined in section 2. We assume that the process is fully automated, i.e. it requires no external resources. Upon the arrival of an order, it has to be entered into the system. This service is provided by the class *order* but it involves also the class *customer* because the respective customer has to be recorded in the order. Note: the fact that the service *enter order* is provided by *order* and not by *customer* is not represented in the EMC. Although it would have been possible to distinguish between providing and otherwise involved classes by employing different arrows,

we decided against this option because during process analysis the modeler is primarily concerned with conceptual issues such as “who has to do with a service?” and not with the exact role a class plays in performing the service. But the latter information, more precisely the assignment of services to classes, is vital for the following design phase and hence should be expressed in the object model (see fig. 9), i.e. while we are still in the analysis phase. According to the EMC, the attributes of *order* are *order id* (e.g. a number) and *items* (a list of ordered items and quantities). These attributes also constitute the skeleton of the respective class definition in the object model (see fig. 9).

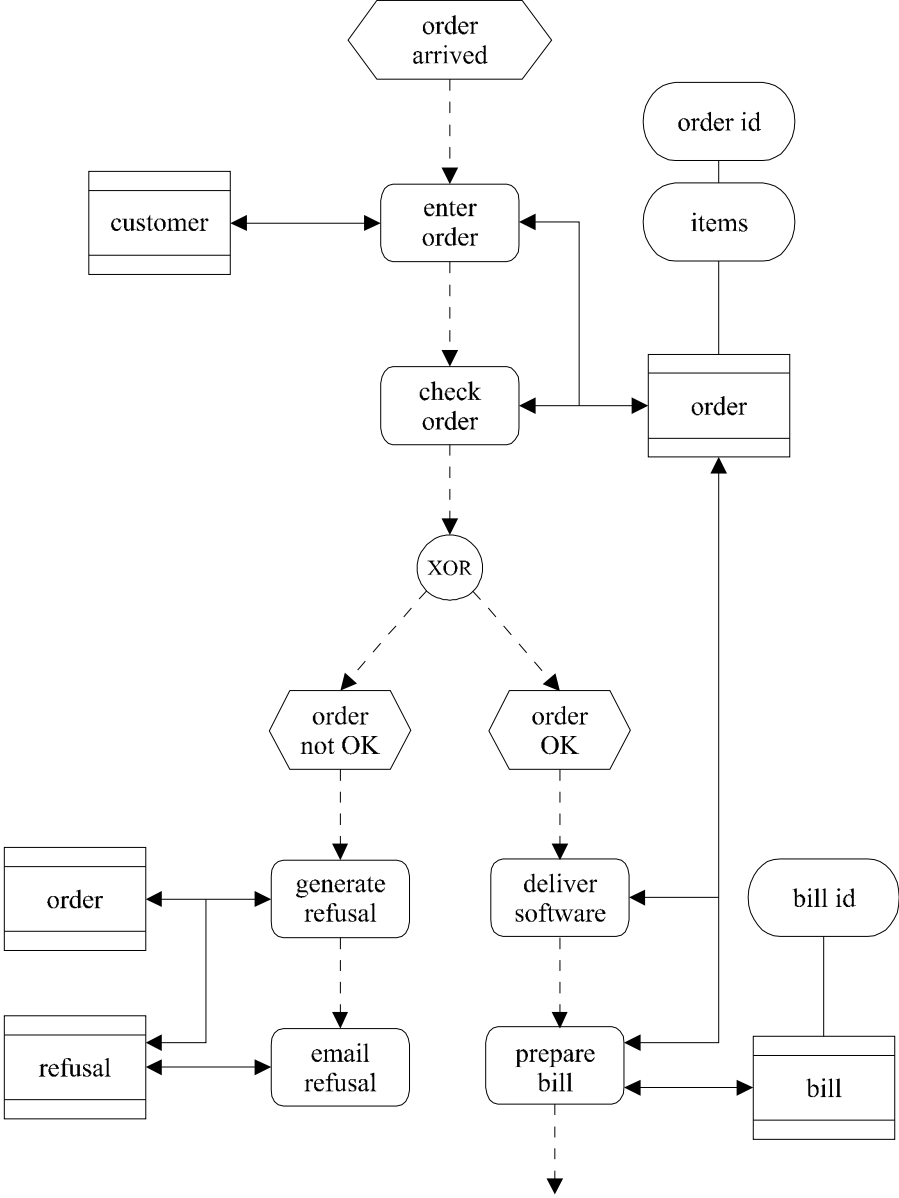


Fig. 8: A part of the EMC for the web application of section 2

After entering the order, it is checked for validity. The outcome of this check is represented by the occurrence of either the event “*order OK*” or the event “*order not OK*”. The XOR split denotes that these events are mutually exclusive. In the case of an invalid order, a refusal of the order is generated and sent via email. The items of a valid order, i.e. the software packages ordered by the customer, are delivered (e.g. via ftp) and the bill is prepared. After this, further processing may occur. Note that no attributes are specified for the class *refusal*. The reason

might be that the modeler could not think of appropriate attributes and hence left this to the later stage of developing the object model.

On the basis of this EMC, an initial object model can be specified without further thinking: it simply consists of all classes and their respective attributes as found in the EMC. After this, the following steps yield a complete object model:

1. *assigning services to objects*: from the classes involved in a service, the one providing it has to be selected. There the service is recorded.
2. *finding missing attributes*: each class is thoroughly examined to check whether attributes have been forgotten e.g. because they are not necessary in the context of the current EMC. This step is best performed after all EMCs for the application lie before us.
3. *identifying potentials for generalization*: classes sharing common attributes or services are potential candidates for generalization inheriting these attributes from a common super-class.
4. *establishing associations between classes*: if more than one class is involved in providing a service, there is usually an association between the involved classes.

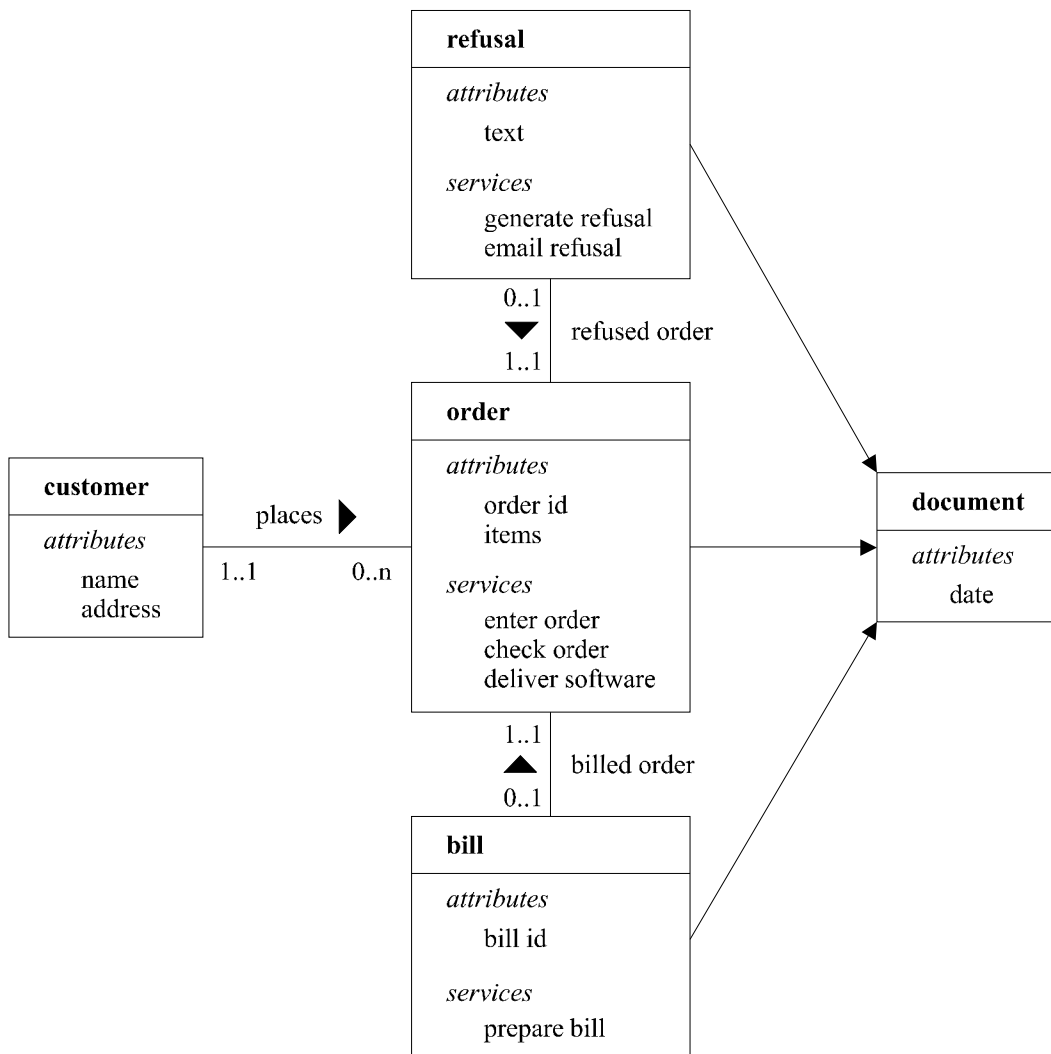


Fig. 9: Object model (structural focus) for the example (in MEMO-OML)

Starting with the initial object model for the EMC of fig. 8, we assign the services to classes as indicated in fig. 9. The EMC gave no attributes for *refusal*. Looking at actual orders, bills and refusals stored in our file cabinet, we find that a refusal contains some explanatory *text* and that all letters carry a *date*. We update the classes accordingly (step 2), and we generalize them to the super-class *document* with attribute *date* (step 3). In the last step, we discover that *customer* and *order* are involved in *enter order*, which leads us to establish an association *places* between them, where a customer can place arbitrarily many orders (0..n) but an order is placed by exactly one customer (1..1). The black triangle indicates the reading direction: *customer places order*. In a similar way, *bill* and *refusal* are connected to *order*.

Please observe that the redundancy of having some of the information present in both EMC and object model (e.g. classes, attributes and services) helps to check inter-model consistency and thus serves model integration (point 1 in section 1).

## 6. CONCLUSION AND OUTLOOK

EMCs provide a way of an integrated analysis of the major aspects of an information system: its structure, its processes and the required resources. Because the underlying paradigm is object-oriented, it enables a seamless transition to object-oriented design and implementation necessary for the development of web-based applications. This is further supported by the unambiguous process semantics of EMCs. In addition, we assume that users already familiar with EPCs will experience few problems in handling EMCs because they resemble each other closely. The process-driven identification of objects helps modelers with less expertise in object-oriented design to create a complete object model, a task that is both highly abstract and challenging even for 'OO gurus'. Put together, these features facilitate an efficient development of web-based applications.

But nevertheless, substantial work remains to be done especially concerning the still blank spots in the MEMO matrix. First of all, we need a language to describe goals to be achieved by the information system, the organization and the company as a whole. More important still, the achievement of the IS goals must be reflected in the respective IS models i.e. the defined goals should somehow guide the development of these models in a goal-achieving direction.

Some preliminary work regarding the strategy and organization levels has been done already, in particular meta models for value chains (SPM) and organizational charts (OSM). However, the inter-level integration remains to be specified. Syntactical integration only requires some kind of common terminology. It can be established by creating a small meta-meta model in the terms of which the meta models for all languages are defined. Semantical integration, on the other hand, is harder to achieve. A possible approach is the object-oriented reconstruction of all models in MEMO-OML.

## REFERENCES

- [CS 94]           Chen, R.; Scheer, A.-W.: *Modellierung von Prozessketten mittels Petri-Netz-Theorie*, IWi-Heft 107, Institut für Wirtschaftsinformatik, Universität Saarbrücken, 1994.
  
- [Frank 97]       Frank, U.: *Enriching Object-Oriented Methods with Domain Specific Knowledge: Outline of a Method for Enterprise Modelling*. Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 4, Universität Koblenz-Landau, 1997.

- [Frank 98] Frank, U.: *The Memo Object Modelling Language (MEMO-OML)*, Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 10, Universität Koblenz-Landau, 1998.
- [KT 97] Keller, G.; Teufel, Th.: *SAP R/3 prozessorientiert anwenden - iteratives Prozess-Prototyping zur Bildung von Wertschöpfungsketten*, Addison-Wesley, Bonn, 1997.
- [LSW 97] Langner, P.; Schneider, Ch.; Wehler, J.: *Prozessmodellierung mit Ereignis-gesteuerten Prozessketten (EPKs) und Petri-Netzen*, WIRTSCHAFTS-INFORMATIK, 39 (1997) 5, S. 479-489.
- [Porter 85] Porter, M.E.: *Competitive Advantage: Creating and Sustaining Superior Performance*. Simon & Schuster, New York, 1985.
- [Ritt 99] Rittgen, P.: *Modified EPCs and Their Formal Semantics*. Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 18, Universität Koblenz-Landau, 1999.
- [Rump 97] Rump, F.: *Erreichbarkeitsgraphbasierte Analyse ereignisgesteuerter Prozessketten*, Technischer Bericht 04/97, Institut OFFIS, Universität Oldenburg, 1997.
- [Scheer 92] Scheer, A.-W.: *Architektur integrierter Informationssysteme*. 2<sup>nd</sup> edition. Springer, Berlin, 1992.
- [Scheer 99] Scheer, A.-W.: *ARIS – Business Process Modeling*. 2<sup>nd</sup> edition. Springer, Berlin, 1999.
- [Speck 98] Speck, M.: *Akzeptanz und Operationalität von EPK in der Modellierungspraxis – ein Erfahrungsbericht aus einem Reengineering-Projekt*, Arbeitskreistreffen Formalisierung der EPK, Münster, 1998-03-17, [http://www-is.informatik.uni-oldenburg.de/~epk/treffen\\_170398/speck.ps](http://www-is.informatik.uni-oldenburg.de/~epk/treffen_170398/speck.ps).