Workflow Modeling for Internet-Based Commerce: An Approach Based on High-Level Petri Nets

Wolfgang Weitz

Institute AIFB, University of Karlsruhe,
D-76128 Karlsruhe, Germany
wolfgang.weitz@aifb.uni-karlsruhe.de

Abstract. In this paper, we discuss the application of a new variant of high-level Petri nets, the so-called SGML nets, for modeling business processes in the area of Internet-based commerce. SGML nets are designed to capture the process of generating and manipulating structured documents based on the international standard SGML. Since the currently dominant document standards on the Internet are HTML (an SGML application) and XML (a subset of SGML), SGML nets offer an elegant way to integrate central aspects of Electronic Commerce applications, such as the generation of online product catalogs, processing of online orders, and electronic document interchange between companies, into a unified formal workflow model. The article gives an introduction to the central concepts of SGML nets and includes an example of their application from the area of online order processing.

Keywords: Internet-based commerce, workflow modeling, Petri nets, structured documents, SGML

1 Introduction

The efficient use of modern information systems combined with wide area communication networks has become a critical success factor for increasing parts of today's modern economies. Electronic Data Interchange (EDI) systems for advanced logistic systems as a prerequisite for just-in-time production, online banking and electronic cash, Internet-based online catalogs and shopping systems, as well as the formation of network-based virtual project teams are only some examples of the different aspects of "Electronic Commerce" (EC). EC may be defined as "the entire collection of actions that support commercial activities on the net" [2].

It is interesting that the same definition can be used to characterize another network-oriented technology that set out to improve the efficiency of business processes, namely *workflow management systems*[1] (WFMS). A typical characteristic of EC systems is the use of a wide area network (WAN) as an interface

[1] See e.g. [4] for an overview on workflow management.

between a company or an individual on the one side and business partners in the "outside world" such as customers or suppliers on the other. In contrast to that, today's workflow management systems predominantly operate in local area networks (LANs) and focus on supporting the business processes *inside* an organization. The rising popularity of Internet technology also in LANs ("Intranets") increasingly removes boundaries between local and global networks from the technical point of view, but concepts for a comprehensive integration of EC components and workflow systems on the modeling and as well as the application level are still missing.

In this paper, we propose the use of a variant of high-level Petri nets, so-called *SGML nets* [9], as a formalism for modeling and executing workflows for Internet-based electronic commerce applications. The paper is organized as follows: The next section discusses the use of Internet technology as a possible solution for current problems with the integration of EC and workflow applications. Section 3 discusses some central concepts and applications of the SGML structured document standard. Section 4 gives an introduction to Petri nets. After that, section 5 presents SGML nets as a method for integrated document and workflow modeling. An example from the area of online order processing follows in section 6. Section 7 contains some concluding remarks.

2 Internet Standards as a Common Platform for EC Applications and Workflow

For a long time, Electronic Commerce was merely used as a synonym for electronic data interchange between companies. Specialized translator programs were necessary to transform the data to be exchanged from the format used in the local information system to some EDI standard format and back (Fig. 1).

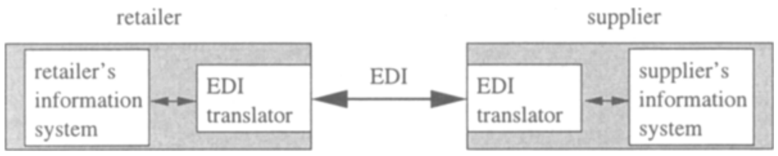


Fig. 1. EDI

A study published in [8] named "the great difficulty and cost of achieving interoperation between the application programs involved" as the main reason for the slow adoption of EDI. Furthermore, software producers complained about "the extreme complexity of the current standards" leading to high training cost for the software developers. This made the resulting EDI products uneconomic or at least unattractive for a majority of smaller and medium companies.

The rapid growth of the Internet has made affordable global network connectivity for large parts of the population in the industrial countries a reality. The possibility to reach people in their homes is a necessary precondition for making the new Electronic Commerce services like online banking or online shopping profitable and thus viable. At the same time, local area networks are more and more being built on open Internet protocols instead of proprietary ones. This increases interoperability between different system platforms and at the same time helps to cut cost by replacing expensive specialized software by inexpensive (or even free) Internet software.

So on the level of networking protocols, a seamless integration of LANs and WANs has already widely taken place, but most business applications are not yet capable to take full advantage of this development. One of the visions that were pursued by the introduction of workflow management systems was the "paperless office", especially to avoid frictions and extra cost induced by the problem of *media discontinuity*. Interestingly enough, a similar problem begins to reappear now in the context of Electronic Commerce applications: The necessity to open formerly closed corporate information systems to "the outside world" reveals the problem of "format discontinuity". Substantial efforts have to be made to interface corporate workflow systems to EC services, each possibly using another data format (Fig. 2 (a)). This is a consequence of the fact that today's workflow systems and models still make the assumption that a business process is always an internal affair of and as such limited to one organization.

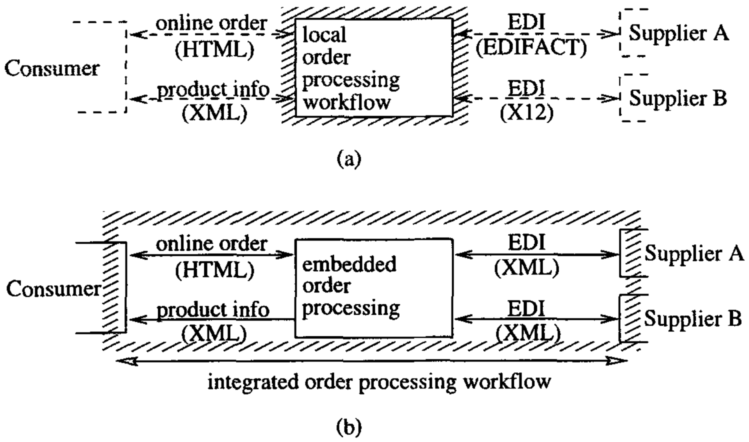


Fig. 2. local vs. integrated workflow modeling

To obtain a better solution it is necessary to recognize that a workflow in the context of Electronic Commerce is *not* local to a single company; quite the opposite is true. It is characteristic for such workflows to consist of multiple parties that have to interact with one another over a network to reach a certain business goal. This shows that it is important for a workflow model in this application domain to capture the local business processes as well as their network-based interaction with the outside world (Fig. 2 (b)), making formerly isolated intra-organizational processes *embedded workflows* in an integrated workflow model.

A possible step in the direction of such a comprehensive integration is to follow the development in the area of network protocols and try to build a workflow model that is capable to integrate Internet document standards like HTML and XML. The SGML nets that will be presented later on are an attempt to provide a formal model as a basis for such a workflow system.

3 SGML-Related Document Standards on the Internet

Structured documents contain both content *and* structure information in an explicit form, which facilitates automated processing. A major international standard in the area of structured documents is SGML [5]. An SGML document consists of two parts, an SGML *Document Type Definition* (DTD) and an *instance* of that DTD carrying the content information. A DTD formally defines a document grammar which specifies a class of conforming document instances. Structure information is added to a text by explicitly marking structural *elements* of the document instance with *tags* that were previously defined in the DTD. The markup of a text element usually takes the form `<TAG>...</TAG>`. Table 1 shows some examples of DTDs and conforming document instances.

What makes SGML interesting in the area of EC is the fact that it is the base technology for the two dominant standard document formats of the popular WWW service on the Internet: HTML and XML [6]. HTML is in fact defined in terms of an SGML DTD, while XML is essentially a subset of SGML. The XML standard includes especially the concept of defining document structure by means of a DTD but leaves out some SGML features that were felt to be irrelevant or too complex for the area of Internet applications.

It is reasonable to expect that the generation and manipulation of HTML and XML documents plays a central role for many Internet-based commerce applications like the generation of online catalogs, online advertising, online shopping systems etc. Furthermore, there are moves to use XML as a standard data format for EC applications. For example, the recently formed XML/EDI group and the Open Trading Protocol (OTP) Consortium are preparing standard proposals for this purpose[2]. Keeping in mind that current EDI implementations tend to be quite expensive, such XML based solutions that allow the use of standard XML editors, browsers and other tools may well turn out to provide a very cost effective and flexible way to build up an Internet-based EC infrastructure.

4 Petri Nets

Petri nets [3] are a well-founded formal notation to model a dynamic system's behavior. Due to their graphical representation and the possibility to introduce hierarchies of nets, even complex and highly parallel models are still manageable.

A *Petri net* is a triple $N = (P, T, F)$, where P is the set of *places* (passive elements) and T the set of *transitions* (active elements). Generally, a place may

[2] See `http://www.xmledi.net` and `http://www.otp.org` for further information.

be interpreted as an object store, while a transition specifies a class of operations on its adjacent places. In the graphical representation, places are depicted by circles and transitions by rectangles. The so-called *flow relation* F is a set of directed arcs. Each arc leads either from a place to a transition or from a transition to a place. An arc leading from a place p to a transition t is called an *input arc* of t, and p is accordingly called an *input place* of t. Analogously, an arc from a transition t to a place p is an *output arc* of t, and p is an *output place* of t. The *vicinity* of a transition is the union of its input places and its output places.

A *marking* of a net N assigns a set of *tokens* to every place in N. It can be interpreted as a global system state. When a transition *occurs*, tokens are removed from its input places and inserted into its output places. The *occurrence rule* defines when a transition is *enabled* (i.e. it *can* occur) and (if so) how tokens in its input and output places are manipulated. As a result, the state of a transition t (enabled or not) is only determined by its vicinity, and conversely t will only manipulate places in its vicinity when it occurs.

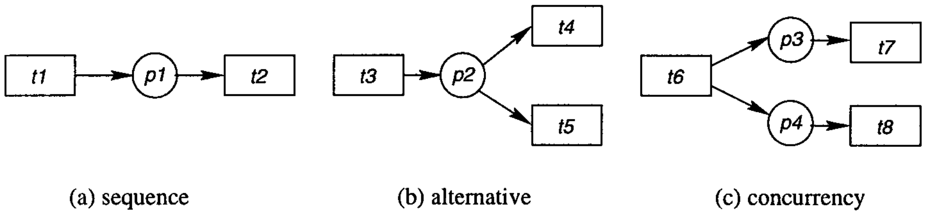


Fig. 3. synchronization primitives

Basic synchronization primitives in Petri net notation are shown in Fig. 3:

- sequence: *t2* consumes tokens from *p1* which may have been previously produced by *t1*.
- alternative: a token in *p2* can be consumed *either* by *t4 or* by *t5*.
- concurrency: *t6* inserts a token into both *p3* and *p4*, which *simultaneously* enables *t7* and *t8*.

Petri net variants where every token has an individual identity (and possibly structure) are called *higher Petri nets.*

5 SGML nets

5.1 Motivation

A necessary precondition for the employment of a workflow management system is that a sufficient degree of structure can be identified in a business process. It is a major objective of workflow modeling to capture and to operationalize that structure. In reality, structured workflows are often accompanied by structured documents, such as order forms, bills, lists of different types, serial letters, receipts, and so on. Structured business processes are often to a large extent controlled by structured documents, since they help to provide necessary information in a compact, complete, non-redundant way and facilitate efficient further processing.

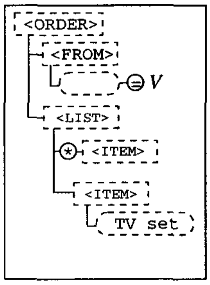
A weak point of today's workflow models is that they cannot capture the (often complex) inner structure of the documents involved. Documents are usually only represented as (unstructured) tokens. This means that decisions based on content and structure of documents as well as concurrent manipulation of a document cannot be specified adequately within the workflow model.

SGML nets [9] are a variant of higher Petri nets. They are designed for modeling workflows that need to create and manipulate structured documents. Every place in an SGML net is assigned an SGML document type definition as its type. A marking of a place is a set of document instances conforming to the place's DTD. An arc in an SGML net is inscribed with a so-called *document template*. Document templates are used to specify operations to identify and manipulate document instances in the adjacent place of the arc. The following sections will give an introduction to basic SGML net concepts.

5.2 Document Templates

Document templates allow a graphical specification of document query and manipulation operations. Following the idea of "query-by-example" [11], a document template is a tree that specifies the structure of the class of document instances to be matched. The matching process always starts at the root element of a document and respects the order of the siblings as well as the specified hierarchic structure. Document templates are not intended to be a full-featured query language for arbitrary databases of structured documents (see [7] for a survey of such languages). Since they are used to inscribe arcs in SGML nets, they can rely on the fact that all document instances to be matched have the same structure, as defined in the DTD of the arc's adjacent place. This makes it possible to use a very readable, graphical notation which matches the graphical nature of Petri nets.

We will now introduce some of the constructs available to specify document templates. A *square box* matches an arbitrary SGML element (including all enclosed subelements). It may be restricted to a certain SGML element type by inscribing the box with an element name. Similarly, an *oval box* represents the textual content (SGML `#PCDATA`) of an element. The *"=" operator* is always accompanied by the name of a variable. It states that the portion of the document matched by that box operator equals the current contents of the variable. To apply a part of the document template repeatedly in the matching process, the *"*" operator* is used. As an example, the document template in Fig. 4 (a) recognizes an `ORDER` document instance where the content of the `FROM` element equals the contents of variable *V* and that has an `ITEM` with content "`TV set`" somewhere in its `LIST` element. Consequently, the example document instance in Fig. 4 (b1) and (b2) are matched by (a), while the document (b3) is not since in this document instance, no `FROM` element occurs *before* the `LIST` element.



(a) document template

```
<ORDER>
  <FROM>
    Carl Customer
  <LIST>
    <ITEM>
      Camera
    <ITEM>
      FAX machine
    <ITEM>
      TV set
  </LIST>
</ORDER>
```

(b1) example document 1

```
<ORDER>
  <FROM>
    Carl Customer
  <LIST>
    <ITEM>
      TV set
  </LIST>
  <TOTALPRICE>
    20000.00
</ORDER>
```

(b2) example document 2

```
<ORDER>
  <LIST>
    <ITEM>
      Camera
    <ITEM>
      FAX machine
    <ITEM>
      TV set
  </LIST>
  <FROM>
    Carl Customer
</ORDER>
```

(b3) example document 3

Fig. 4. document template and example document instances

5.3 Arc Inscriptions

In section 4 we saw that generally in Petri nets, tokens are removed from the input places and new tokens are inserted into the output places when a transition occurs. Since the tokens in SGML nets are document instances with a (possibly complex) inner structure, we are interested to be able to specify operations not only on documents as a whole, but also on substructures of a document. For these operations it is necessary to make sure that any document instances in an output place can be identified unambiguously. We therefore require that a subset of the elements in a place's DTD is declared a *key* for this DTD and that any two document instances in the marking of this place must disagree in the content of at least one of the key elements (see Table 1 for examples).

In the document templates of input and output arcs, box operators drawn with a solid instead of a dashed line are used to indicate the location of an intended modification. In the case of input arcs, an SGML element matched by a square box with a solid line will be *discarded* when the transition occurs. For example, a document instance with a root element of type ORDERFORM can be matched with the simple template $\boxed{\texttt{<ORDERFORM>}}$ and will be discarded entirely as soon as the respective transition occurs. Similarly, arbitrary substructures in a document can be marked for removal. As we will see later on, the definition of the occurrence rule for SGML nets will prevent possible violations of the respective places DTD as a result of such an operation.

For the inscription of output arcs, we need to distinguish two situations: additionally to an operation for *creating* a new token in an output place as known from other Petri net variants, we need a way to specify an *insertion* of new elements into an already *existing* document instance. To guarantee unambiguity, the document templates must contain at least the key elements of the respective output place's DTD.

As before, boxes drawn with a solid line indicate a modification operation, which is in this case the creation of new SGML elements. Consequently, a document template to create a whole new document instance consists entirely of boxes with solid lines. An insertion into an existing document instance is simply achieved by specifying a document template that identifies the document instance to be modified and augmenting that template with solid boxes to define the elements to be inserted. Fig. 5 gives an example: transition "authorize_order" is used to create a new **ORDERFORM** document, transition "add_item" inserts a new **ITEM** (variable *IT*) at the head of the **LIST**.

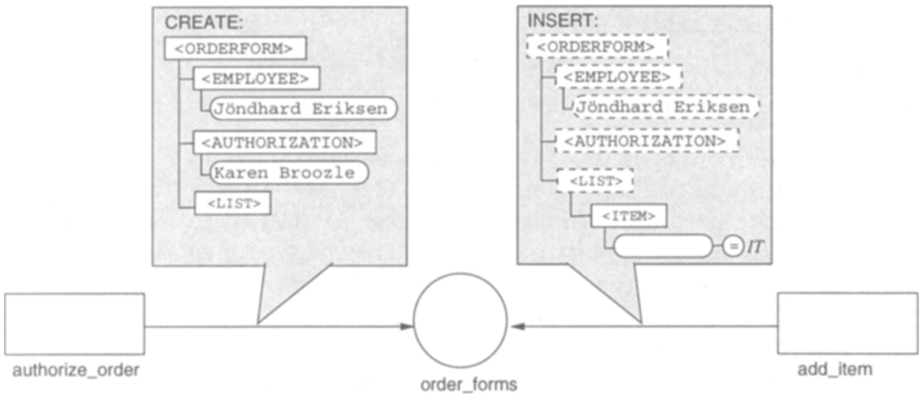


Fig. 5. arc inscriptions

5.4 Occurrence Rule

We can now define the occurrence rule for SGML nets. A transition t in an SGML net is *enabled* for a given marking and an instantiation of the variables in t's and its adjacent arcs' inscriptions if the following conditions are satisfied:

- Every input place contains a document instance which can be matched with the document template in the inscription of the arc leading from p to t *and* extracting elements as specified by the arc's inscription will not cause a violation of p's DTD.
- Transitions may be inscribed with logical expressions. These may include variables that occur in the inscriptions of the transition's input and output arcs. This transition inscription has to evaluate to "true". Note that reading data from "external" sources (user input, sensor data, etc) as well as operations to manipulate content (rather that structure) can be modeled by introducing a predicate for the intended purpose and using it in the transition inscription (see transition submit_online_order in Fig. 7 for an example).
- For each output arc of t specifying an "insert" operation, this insertion into the matched document instance must not lead to a violation of the adjacent place's DTD.
- For each output arc of t defining a "create" operation, the resulting document instance must conform to the adjacent place's DTD, and there must not already exist a document instance with matching key element values.

Only if all these conditions are met, the transition can occur and the manipulation of the document instances in the places in its vicinity is performed as specified by the transition and arc inscriptions. Note that starting from a valid initial marking of the net, the definition of the occurrence rule *guarantees* that no invalid document instance can possibly be generated in any following state. In every reachable system state, all document instances always conform to their respective places's DTD.

5.5 Special Operators

There are two more operators that are used in an example later on, negation ($\ominus$) and the so-called "zap" operator ($\oslash$). The negation operator succeeds if the subordinate document template *cannot* be matched (negation as failure). This document template must not specify any insertion operations, and the use of a variable in a negated template is only allowed if it is "safe", i.e. if the same variable does also occur in a non-negated template belonging to the same transition. In Fig. 7, negation is used in the inscription of arc (stocks, pass_on_order) to find out if there is *no* document with type PRODUCT and a certain NAME is currently in place stocks.

The operator $\oslash$ permits a controlled violation of a document's DTD in the inscription of an input arc. Normally, a transition cannot occur if this would lead to the violation of a place's DTD. The operator $\oslash$ lifts this restriction for its subordinate document template but results in the *whole* document instance to be discarded when the transition occurs. This definition may seem a bit exotic at a first glance, but it is very useful for the common task of iterating over a sequence of elements, e.g. the ITEMs of a LIST. Although the functionality of $\oslash$ can be simulated by adding a transition that handles the case leading to the violation of the DTD separately, using $\oslash$ can make SGML net models readable. See the inscription of arc (orders,ship_item) in Fig. 7 for an example: by its DTD (Table 1), an ORDER form must contain a LIST element with at least one ITEM (which is certainly a quite reasonable requirement from the document modeling point of view). Consequently, without the $\oslash$ operator, the transitions ship_item and pass_on_order will extract all ITEMs from an ORDER *except* the last one. The use of the $\oslash$ operator now permits also the last ITEM to be processed as the others and after that, the now completely processed ORDER document is discarded.

6 Example: Online-Order Processing

To illustrate the concepts introduced in the previous sections, we will now discuss an example modeling an "online order processing" workflow. Figure 7 shows the corresponding SGML net model and Table 1 contains the necessary document type definitions as well as example document instances. Due to space limitations, only the embedded workflow at a computer retailer's local branch is modeled in detail, while for the other two participating parties (customer and central warehouse), only the interfaces are included.

Place	document type definition (DTD)	example document instance
orders	`<!DOCTYPE order [` `<!ELEMENT order - - (from,list)>` `<!ELEMENT from - o (#PCDATA)>` `<!ELEMENT list - - (item+)>` `<!ELEMENT item - o (#PCDATA)>` `<!KEY from>` `]>`	`<ORDER>` `<FROM>` `Betty Buyer` `<LIST>` `<ITEM>Mousepad XL` `<ITEM>SCSI Cable` `<ITEM>K3,3 CPU` `</LIST>` `</ORDER>`
stocks, stock-receipt	`<!DOCTYPE product [` `<!ELEMENT product - - (name,serialno)>` `<!ELEMENT name - o (#PCDATA)>` `<!ELEMENT serialno - o (#PCDATA)>` `<!KEY name,serialno>` `]>`	`<PRODUCT>` `<NAME>` `Monitor M33` `<SERIALNO>` `M35647X97` `</PRODUCT>`
incoming requests	`<!DOCTYPE request [` `<!ELEMENT request - - (from,customer,product)` `<!ELEMENT from - o (#PCDATA)>` `<!ELEMENT customer - o (#PCDATA)>` `<!ELEMENT product - o (#PCDATA)>` `<!KEY from,customer,product>` `]>`	`<REQUEST>` `<FROM>` `Branch Main St.` `<CUSTOMER>` `Betty Buyer` `<PRODUCT>` `K3,3 CPU` `</REQUEST>`
delivery notes	`<!DOCTYPE deliverynote [` `<!ELEMENT deliverynote - - (customer,list)>` `<!ELEMENT customer - o (#PCDATA)>` `<!ELEMENT list - - (item+)>` `<!ELEMENT item - o (#PCDATA)>` `<!KEY customer>` `]>`	`<DELIVERYNOTE>` `<CUSTOMER>` `Betty Buyer` `<LIST>` `<ITEM>Mousepad XL` `<ITEM>SCSI Cable` `</LIST>` `</DELIVERYNOTE>`

Table 1. places in Fig. 7 and their corresponding DTDs

Customers may submit an order form over the Internet to a computer retailer's local branch. This is modeled by transition submit_online_order which creates a new document of type **ORDER** (Fig. 1 (a)) in place orders. Now, the ordered items are checked for availability. Here, two cases are possible:

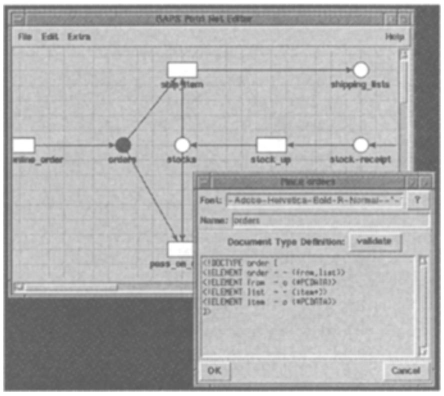
– Transition ship_item is enabled for an item *IT* if *IT* is listed in the order *and* if that item is on stock, i.e. there is a corresponding **PRODUCT** document in place stocks. In this case, the **item** with content *IT* is discarded from the order, and the matched **PRODUCT** is removed from stocks. At the same time, a new **ITEM** with content *IT* is inserted at the end of the **LIST** element into the **DELIVERYNOTE** document for customer *CU* if such a document already exists. Otherwise, a new **DELIVERYNOTE** for this customer is created.

– If, on the other hand, the ordered item *IT* is currently *not* on stock at the local branch, a **REQUEST** document is created, containing the name of the branch as well as that of the customer and the ordered product. Transition pass_on_order models the creation and EDI transmission of such a request to the company's central warehouse (place incoming_requests), which then in turn may initiate the delivery of this type of product to the local branch (transition deliver) in order to complete its stocks and also to send the product directly to the consumer (due to a lack of space, this and other processes in the "central warehouse" are not explicitly modeled in our example).

Note that in either case, a ORDER document instance is discarded as soon as the last ITEM is processed, due to the ⊘ operator in the inscriptions of the arcs starting at place orders.

7 Conclusion and Outlook

In this paper, we presented SGML nets as workflow modeling formalism for Internet-based commerce applications. SGML nets combine the solid formal foundation of Petri nets with a possibility to directly generate and manipulate complex HTML and XML document structures, as needed for the generation of online catalogs, online ordering systems or electronic data interchange. Due to their formal semantics, SGML nets can be executed by a net interpreter program, which can be used as the core of a workflow engine for distributed, Internet-based commerce applications.
The ability to specify access to document structure inside the workflow model makes it possible to analyze business processes on a finer level of granularity. One interesting application is the formal analysis of the flow of classified *parts* of documents in order to verify access and security policies in an organization.



```
<!DOCTYPE order [
<!ELEMENT order - - (from,list)>
<!ELEMENT from  - o (#PCDATA)>
<!ELEMENT list  - - (item+)>
<!ELEMENT item  - o (#PCDATA)>
]>
```

Fig. 6: SGML Net Editor

The example in section 6 showed that SGML net diagrams may become quite large even for small examples, which means that appropriate tool support is necessary to use them effectively. Work is on the way to extend GAPS++ [10], for use with SGML nets. GAPS++ is an open Petri net modeling and simulation environment, based on a CORBA compatible distributed object technology.

Acknowledgements
The author would like to thank Jörg Desel, Andreas Oberweis, Wolffried Stucky, and the anonymous referees for their helpful comments on an earlier version of this work.

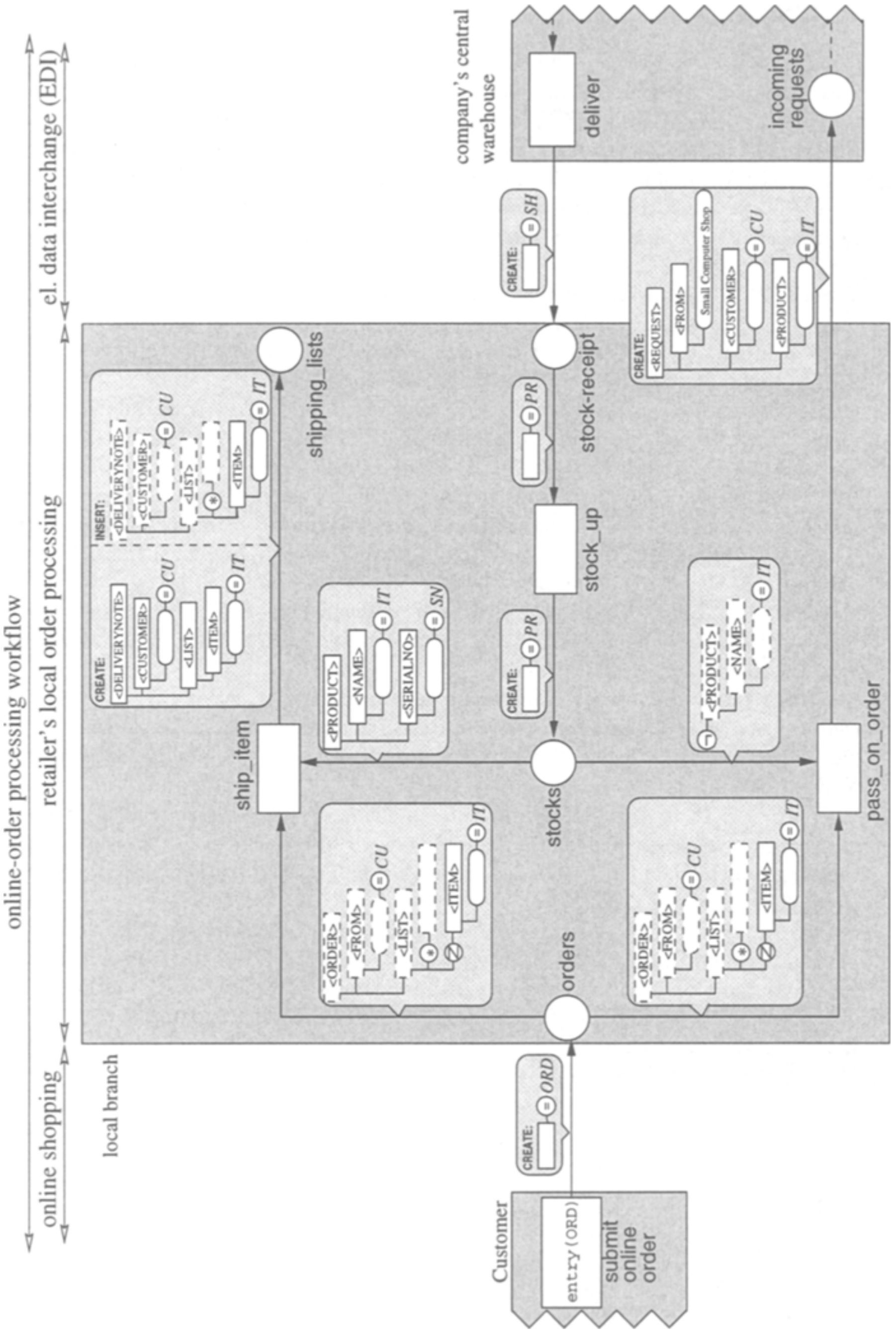


Fig. 7. order processing

References

1. van der Aalst, W. M. P.: Three good reasons for using a petri-net-based workflow management system. In: S. Navathe and T. Wakayama (eds.), Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96) (Camebridge, Massachusetts, November 1996), 179–201.
2. Adam, N. R., Yesha, Y.: Electronic commerce: An overview. In: Adam, N. R., Yesha, Y. (eds.): Electronic Commerce: Current Research Issues and Applications, Lecture Notes in Computer Science, Vol. 1028. Springer-Verlag, Berlin Heidelberg New York (1996) 5–12
3. Brauer, W., Reisig, W., Rozenberg, G. (Eds.): Petri nets, central models and their properties. Advances in Petri nets 1986, part I. Proceedings of an advanced course (Bad Honnef, September 8–19, 1986). Lecture Notes in Computer Science, Vol. 254. Springer-Verlag, Berlin Heidelberg New York (1987)
4. Georgakopoulos, D., Hornick, M. F., Sheth, A. P.: An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and Parallel Databases, Vol. 3, No. 2 (1995) 119–153
5. ISO 8879: Information processing – text and office systems – standard generalized markup language (SGML). International Organization for Standardization (1986)
6. Khare, R., Rifkin, A.: XML: A door to automated web applications. IEEE Internet Computing, Vol. 1, No. 4 (1997) 78–87
7. Navarro, G., Baeza-Yates, R.: Proximal nodes: A model to query document databases by content and structure. ACM Transactions on Information Systems, Vol. 15, No. 4 (Oct. 1997) 400–435.
8. Steel, K.: The standardisation of flexible EDI messages. In: Adam, N. R., Yesha, Y. (eds.): Electronic Commerce: Current Research Issues and Applications, Lecture Notes in Computer Science, Vol. 1028, Springer-Verlag, Berlin Heidelberg New York (1996) 13–26.
9. Weitz, W.: SGML-Nets: Integrating document and workflow modeling. In: Ralph H. Sprague, Jr. (ed.), Proc. 31st Annual Hawai'i International Conference on System Science (Kohala Coast/HI, USA, January 6–9, 1998), IEEE Computer Society Press, Los Alamitos, 185–194.
10. Weitz, W., Stucky, W.: GAPS++: An Internet-based Petri net modeling and simulation architecture. In: P. A. Fishwick, D. R. C. Hill, R. Smith (eds.), Proc. 1998 International Conference on Web-Based Modeling and Simulation (San Diego/CA, USA, January 11–14, 1998), Society for Computer Simulation, San Diego, 131–136
11. Zloof, M.: Query by example. In: Proceedings of the National Computer Conference (1975), Vol. 44, AFIPS Press