1 THREE GOOD REASONS FOR USING A PETRI-NET-BASED WORKFLOW MANAGEMENT SYSTEM

W.M.P. van der Aalst

Eindhoven University of Technology,
Department of Mathematics and Computing Science,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
wsinwa@win.tue.nl

Abstract: Currently, the Dutch Customs Department is building a nationwide information system to handle all kinds of declarations related to the import and export of goods. For this purpose the Petri-net-based Workflow Management System (WFMS) named COSA has been selected. During the selection process, it turned out that there are several reasons for insisting on a Petri-net-based WFMS. The three main reasons for selecting a Petri-net-based WFMS are discussed in this paper. In our opinion these reasons are also relevant for many other projects involved in the selection or implementation of a WFMS.

1.1 INTRODUCTION

At the moment more than 250 Workflow Management Systems (WFMSs) are under development. This signifies that the term 'workflow management' is not just another buzzword. The phenomenon workflow management will have a tremendous impact on the next generation of information systems [HL91, Kou95, Sch96, AH97]. To appreciate the relevance of workflow management one should look back in history. In the sixties an information system was composed of a number of stand-alone applications. For each of these applications an application-specific user interface and database system had to be developed,

i.e. each application had its own routines for user interaction and data storage and retrieval. In the 70-ties data was pushed out of the applications. For this purpose Database Management Systems (DBMSs) were developed. By using a DBMS, applications were freed from the burden of data management. In the 80-ties a similar thing happened for user interface management. The emergence of User Interface Management Systems (UIMSs) enabled application developers to push the user interaction out of the applications. In our opinion WFMSs are the next step in pushing generic functionality out of the applications. The 90-ties will be marked by the emergence of workflow software, allowing application developers to push the business procedures out of the applications. Figure 1.1 shows the phenomenon workflow management in a historical perspective.

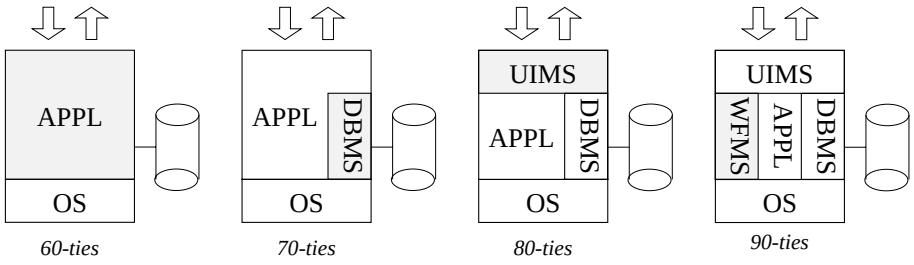

Figure 1.1 WFMSs in a historical perspective.

The Workflow Management Coalition (WFMC), founded in 1993, is an international organization whose mission is to promote workflow and establish standards for WFMSs. In January, 1995 the WFMC released a glossary which provides a common set of terms for workflow vendors, end-users, developers and researchers [WFM96]. In this glossary a WFMS is defined as being a system that completely defines, manages and executes workflow processes through the execution of software whose order of execution is driven by a computer representation of the workflow process logic. Instead of the term 'workflow process logic' we prefer the term 'business logic' to reflect the fact that the configuration of the WFMS is subordinate to the underlying business processes.

The benefits of a WFMS are comparable to the benefits of an UIMS or a DBMS. Flexibility, integration of applications and a reduction in development costs are the incentives for using a WFMS. The importance of workflow management was also recognized by the Dutch Customs Department when they started a project for the development of a nationwide information system for the handling of Cus-

toms declarations. The project was named after the name of the information system under development: *Sagitta-2000*. Since the regulations with respect to Customs declarations are very complex and subject to change, the flexibility and the capability to integrate applications of workflow management software were the prime incentives to start the selection of a WFMS for the realization of Sagitta-2000.

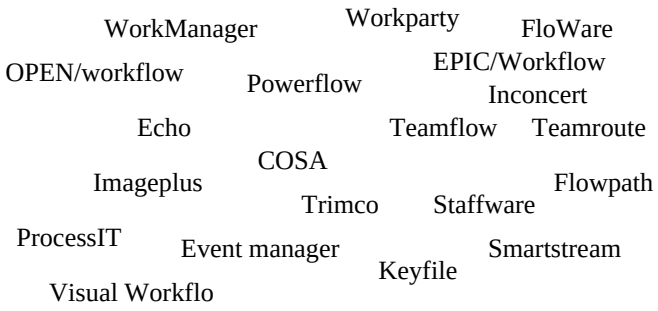

Figure 1.2 Some of the leading WFMSs.

During the selection process some of the leading WFMSs were evaluated using a list of generic selection criteria and a list of functional requirements specific for the Sagitta-2000 project (see Figure 1.2). In the beginning the results were quite disappointing. First of all, the selection process was hampered by the fact that, despite the efforts of the Workflow Management Coalition, standardization is lacking. Secondly, most of the leading WFMSs fail to represent the business processes of the Dutch Customs Department in a natural manner. Many WFMSs have restrictions with respect to the nesting and/or mixing of parallelism and alternative routing. Moreover, most of the WFMSs do not allow for the explicit modeling of states. As result it is not possible to handle triggers and external choices properly (See Section 1.4). Thirdly, only a few WFMS could be used in the technological environment (i.e. hardware platform, operating system, DBMS, etc.) of the Dutch Customs Department. During the selection process it dawned upon the people involved in the Sagitta-2000 project that a Petri-net-based WFMS could meet all of the functional requirements needed. It turned out that there are at least three good reasons for selecting a Petri-net-based WFMS:

- Formal semantics despite the graphical nature.
- State-based instead of event-based.

- Abundance of analysis techniques.

On the basis of these observations the Petri-net-based WFMS COSA [SL96] was selected for the local hardware platform. For the central workflow engine of Sagitta-2000 there were no suitable candidates. Therefore, the Dutch Customs Department decided to start building a proprietary workflow engine based on the Petri net formalism. We would like to emphasize that these decisions were based on objective arguments. The people responsible for the selections were not biased towards Petri nets. In fact, most of them had no prior knowledge of Petri nets.

The reasons for selecting a Petri-net-based WFMS are quite universal and certainly not specific for the Dutch Customs Department. They hold for most situations where the introduction of a WFMS is considered. In the remainder of this paper we introduce the Sagitta-2000 project followed by a discussion on each of the three main reasons to use a Petri-net-based WFMS.

1.2 SAGITTA-2000

The Sagitta-2000 project started in 1994. The goal of this project is to develop a nationwide information system for the processing of all kinds of Customs declarations. The processing of a Customs declaration is a very complex process which is subject to change. Activities that are needed to handle a declaration are typically related to the registration, checking and control of movements of communal goods. For some types of declarations more than 50 activities can be identified.

At the moment the handling of Customs declarations is partly automated. A number of legacy systems support the management of data related to the processing of declarations. The management of the processes is hardly supported at all. Paper documents form the pivot on which the processing of Customs declarations turns. As a result, the processes are hard to manage and service to the customer is poor. Moreover, the legacy systems are poorly integrated and form a patchwork which reflects the history of Dutch Customs regulations. The goal of the Sagitta-2000 project was to build a flexible well-integrated information system which also supports and manages the process itself.

One of starting points of the Sagitta-2000 project is the separation of information logistics and the implementation of Customs tasks. From the start, it was clear that it would be nice to use a WFMS for the information logistics. By using a WFMS as the basis for Sagitta-2000, it should be easy to accommodate the system to the continual changes of the regulations with respect to

Customs declarations. Flexibility, maintainability and the ability to integrate applications were the keywords that served as a stimulus for using a WFMS.

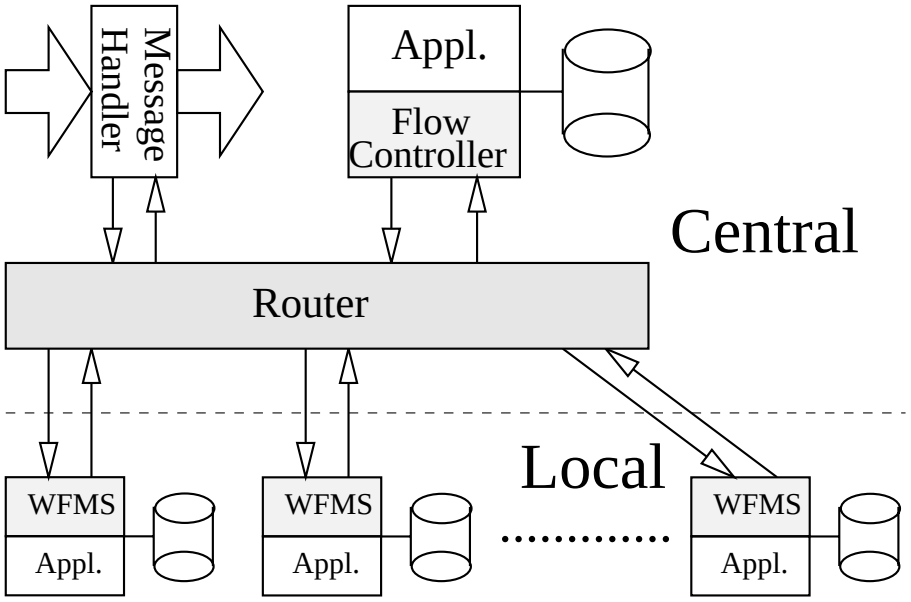

Figure 1.3 The architecture of Sagitta-2000.

Sagitta-2000 will be a distributed information system, composed of a central system in Apeldoorn and dozens of local systems (one for each Customs office). Figure 1.3 shows the architecture of Sagitta-2000. Messages which relate to Customs declarations are received by a *Message Handler*. The Message Handler translates these EDI-messages to an in-house format. At any moment a case, i.e. a Customs declaration, is being handled by one of the local platforms or by the central system located in the city Apeldoorn. The *Router* sends an incoming message which relates to a specific case (Customs declaration) to the proper location. Customs information about declarations is stored in a central database. About 50% of the cases do not require user interaction and are handled completely automatically by the central system. The other 50% require user interaction and need to be handled at a specific Customs office. Cases which require interaction with a Customs officer are partly handled by the central system and partly by one of the local systems, i.e., a case is transferred from the central system to one of the local system the moment user interac-

tion is required. The Router system takes care of these case transfers and the routing of messages. The central system which takes care of the processing of tasks for specific cases is split into two parts: (1) a *Flow Controller* and (2) a set of *applications* for the execution of Customs tasks. The Flow Controller is a system which takes care of the case logistics, i.e., it decides when to execute which task for a specific case using control information about the case. The functionality of the Flow Controller is comparable to the 'engine' in a WFMS. The Flow Controller initiates tasks by starting the proper applications. Note that only the applications can access the Customs information about declarations. The local platform has an architecture which is similar to the central system. Each of the local systems is also split into two parts: (1) a *WFMS* and (2) a set of *applications* for the execution of Customs tasks. The WFMS takes care of the case logistics by initiating required tasks. In addition the WFMS assigns interactive tasks to Customs officers. Note that the applications executed at the local platform which require user interaction cannot be executed by the central system. For performance reasons only, Customs information about declarations handled locally is temporarily stored in a local database.

Performance issues are a constant point of attention for Sagitta-2000. The estimated number of declarations per year is more than 10.000.000. Moreover, there will be days on which more than 70.000 declarations have to be handled! The number of Customs officers using Sagitta-2000 will be more than 5000. These figures show that the Sagitta-2000 project is a very ambitious project. At the moment nearly 100 persons are involved in the development of Sagitta-2000.

The logistics part of the information system is the crux of the Sagitta-2000 project. The workflow procedures used by the local WFMS and the central Flow Controller need to be a good reflection of the business processes at hand. Although there are, from a technical point of view, a lot of differences between the local and the central platform, the business processes are platform-independent. Therefore, the Sagitta-2000 project started with the modeling of the business processes. Each business process describes which tasks need to be executed in order to handle a Customs declaration (case). A business process also specifies which tasks can be executed in parallel, whether there are alternative tasks or iterations. For this purpose the Sagitta-2000 team developed a diagraming technique which is based on Petri nets and inspired by the Glossary of the WFMC [WFM96]. Figure 1.4 shows an example of a business process[1]. Tasks are modeled by rectangles. For each task it is specified how it is triggered and which application is initiated by the triggering of this task. Tasks are connected by triangles named 'work-stores'. Work-stores specify enabling

conditions for a task to be triggered. In Petri-net terms work-stores correspond to places and each task corresponds to a small network taking care of synchronization, triggering and the execution of the corresponding application. The diagraming technique has been used to model the business processes relevant to Sagitta-2000 successfully.

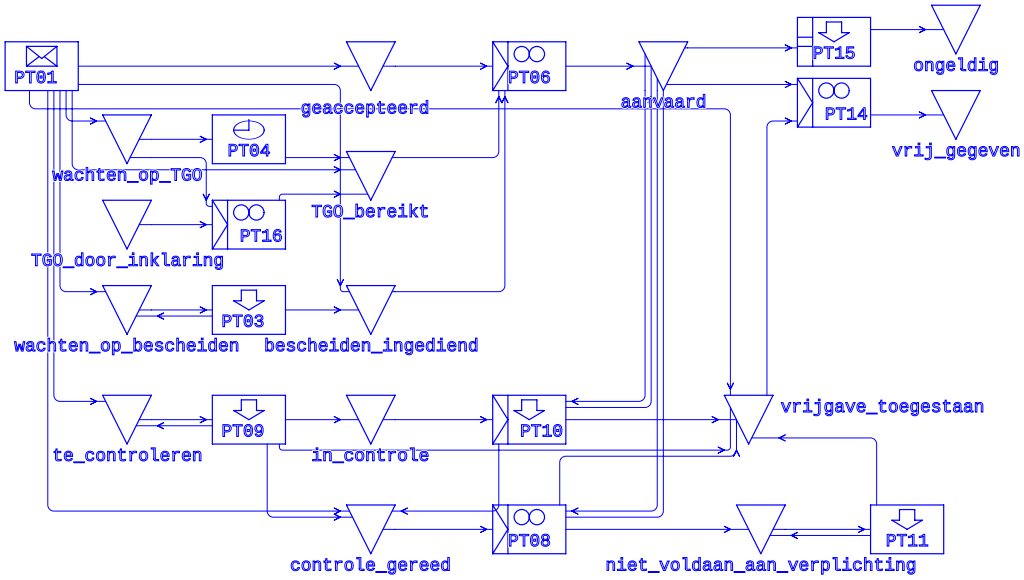

Figure 1.4 The business process 'basis-aangifte' (in Dutch).

Parallel to the definition of the business processes the technical infrastructure (hardware/software) has been selected. For the central system an IBM ES9000 mainframe will be used. The DBMS used for the central system is DB2 and the Message Handler, the Router and the Customs applications are being implemented using Cobol, CICS and DB2. Since suitable workflow products are missing for the mainframe, the Flow Controller is also implemented using Cobol, CICS and DB2.

For the local platform a client/server architecture is used. The server is a HP 9000 connected to dozens of client PCs using a Novell network. The server runs under UNIX and the clients are using Windows. The DBMS used for the local platform is Sybase. For the interactive applications the combination of Powerbuilder, C++ and Sybase is used. The Open Server for CICS is used

to exchange Customs data between the central DB2 database and the local Sybase database. The exchange of cases will be handled by the IBD, a service offered by the Dutch PTT. Figure 1.5 shows an overview of the chosen technical infrastructure.

Given the technical infrastructure and the functional requirements, the Sagitta-2000 team had to select a WFMS for the local platform. Based on the technical infrastructure we made a first selection. The WFMS should be able to operate in a client/server environment (UNIX/Windows). Moreover, the WFMS should be able to communicate with Powerbuilder and use Sybase for data management. Using these technical requirements and the vendor profiles we made the following shortlist: OPEN/Workflow (Wang), Workparty (Siemens), COSA (Software Ley) and Visual WorkFlo (FileNet/Olivetti). We used this shortlist and the functional requirements as the basis for a further selection. After visiting some of these vendors and looking at the brochures of many other WFMSs we became very pessimistic. Most of the WFMSs did not meet the functional requirements needed. Business processes which were easy to formulate in the Petri-net-based diagrams were difficult to implement using most of the WFMSs we evaluated. Fortunately, we also selected the Petri-net-based WFMS COSA [SL96]. COSA met all the functional requirements. In fact, it was possible to translate the Petri-net-based diagrams for the business processes semi-automatically into workflow procedures for COSA. At that moment we realized that the use of a Petri-net-based WFMS for the local platform was essential for the success of Sagitta-2000.

At the same time we tried to select a WFMS for the central system. Unfortunately, we discovered that workflow management software is focussed on client/sever technology. The only interesting product was Early Cloud's Message-Driven Processor (MDP), a message-oriented middleware workflow product which runs on mainframes (MVS/CICS). A more thorough investigation showed that MDP was not a suitable candidate for the Flow Controller. Therefore, the Dutch Customs Department decided to develop a tailor-made workflow engine using Cobol, CICS and DB2. At the moment this workflow engine is being built. Based on the success of the Petri-net-based approach for the modeling of the business processes and the local WFMS, the Flow Controller is also based on Petri nets.

For the Dutch Customs Department and in particular the Sagitta-2000 project, the selection of Petri-nets as a vehicle for modeling and implementing the central and local workflow turned out to be very promising. Moreover, we are convinced that the use of a Petri-net-based WFMS has a number of real advan-

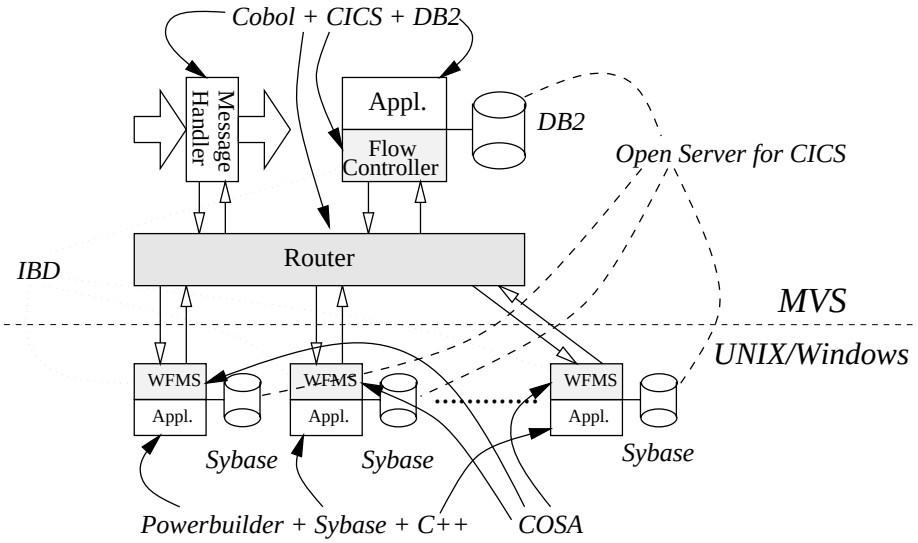

Figure 1.5 The technical infrastructure of Sagitta-2000.

tages which are not specific to the Sagitta-2000 project. Any other information system which supports complex business processes can benefit from the use of a Petri-net-based WFMS. Therefore, we present three universal and solid reasons for using a WFMS based on Petri nets.

1.3 REASON 1: FORMAL SEMANTICS DESPITE THE GRAPHICAL NATURE

The first reason for using a Petri-net-based WFMS, is the fact that business logic can be represented by a formal but also graphical language. The semantics of the classical Petri net and several enhancements (color, time, hierarchy) have been defined formally [Hee94, Jen92, Mur89, Rei85]. In this section we will show that a Petri net can be used to model the primitives identified by the Workflow Management Coalition [WFM96]. These primitives are also present in today's WFMSs. To discuss these workflow primitives we start by introducing some terminology.

The objective of a WFMS is to handle *cases* successfully. Examples of cases are insurance claims, orders, mortgages, tax-returns and loans. A *task* is a piece of work whose execution contributes to the completion of a business process.

Synonyms for task are process activity, logical step and work element. A *task instance* is a task that needs to be executed to handle a specific case. Task instances are executed by *resources*. A resource is a human, an application or a combination of a human and one or more applications. Synonyms for resource are actor or participant. The capabilities of a resource are given by a set of *roles*. Each task requires a specific role. Roles are used to map task instances to resources. A *workflow procedure* defines a partial ordering of tasks to handle cases of a specific type. A *workflow process* definition comprises a workflow procedure, a set of resources and a strategy to map task instances to resources.

One should clearly differentiate between workflow process definition and workflow process execution. Workflow process definition is concerned with the design of tasks, procedures, roles and resources using a design and analysis tool. Workflow process execution is concerned with the enactment of cases and task instances using a workflow engine.

Features of a Petri-net-based WFMS are most prominent in the design and analysis fase. Therefore, we concentrate on the workflow process definition.

Figure 1.6 shows how the six workflow primitives identified by the Workflow Management Coalition [WFM96] can be mapped onto Petri nets. Tasks are mapped onto transitions and causal relations are modeled by places. Transition *t1* models the synchronization of two subflows (AND-join). Transitions *t21* and *t22* model an OR-join: two subflows are merged into one subflow. Transition *t3* models an AND-split: a subflow is split into two parallel subflows. Transitions *t41* and *t42* model an OR-split: a selection is made between two alternative branches. Iteration can be modeled by adding a feedback transition (*t52*). Connecting two transitions (*t61* and *t62*) by means of an intermediate place, results in two sequential tasks.

The state of a case c is given by the distribution of tokens corresponding to c over the places in the Petri net. To distinguish between tokens corresponding to different cases we use a high-level Petri net model [Jen92, Hee94] extended with color. The color or value of a token contains information about the case the token belongs to and some additional information (e.g. routing parameters, due-date, responsible or preferred resource). Note that each transition which models an AND-join requires a precondition to prevent tokens corresponding to different cases from being mixed.

To illustrate the use of Petri nets for the modeling of workflow procedures, we consider the processing of complaints.[2] A complaints desk handles complaints of customers about the products produced by the fictitious Company X. Each complaint is registered before it is classified. Depending on the classification of the complaint, the complaint is ignored, a letter is sent to the customer or

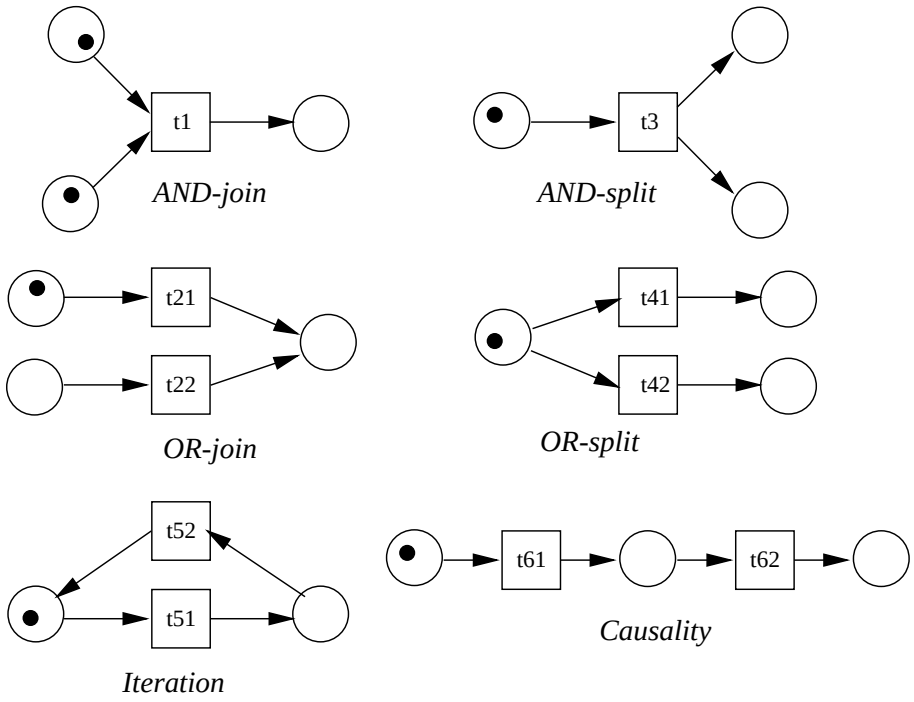

Figure 1.6 Workflow primitives.

an inquiry is started. The inquiry starts with a consultation of the department involved, followed by a discussion with the customer and the management of the department (in parallel). Based on this inquiry the necessary actions are taken. Finally, the dossier is filed. At any time between the registration of a complaint and the moment the complaint is filed, the customer may inform about the status of the corresponding complaint. Figure 1.7 shows a specification of the workflow procedure used to process complaints. Even for this simple example we need all the primitives identified by the WFMC.

The workflow primitives shown in Figure 1.6 are used to define workflow procedures. However, to complete the definition of a workflow process we have to add another dimension: the dimension which takes care of the mapping of tasks to resources. The dimension of workflow procedures and the dimension of resource management are orthogonal and therefore difficult to visualize in one Petri net. Nevertheless, it is possible to model workflow procedures and

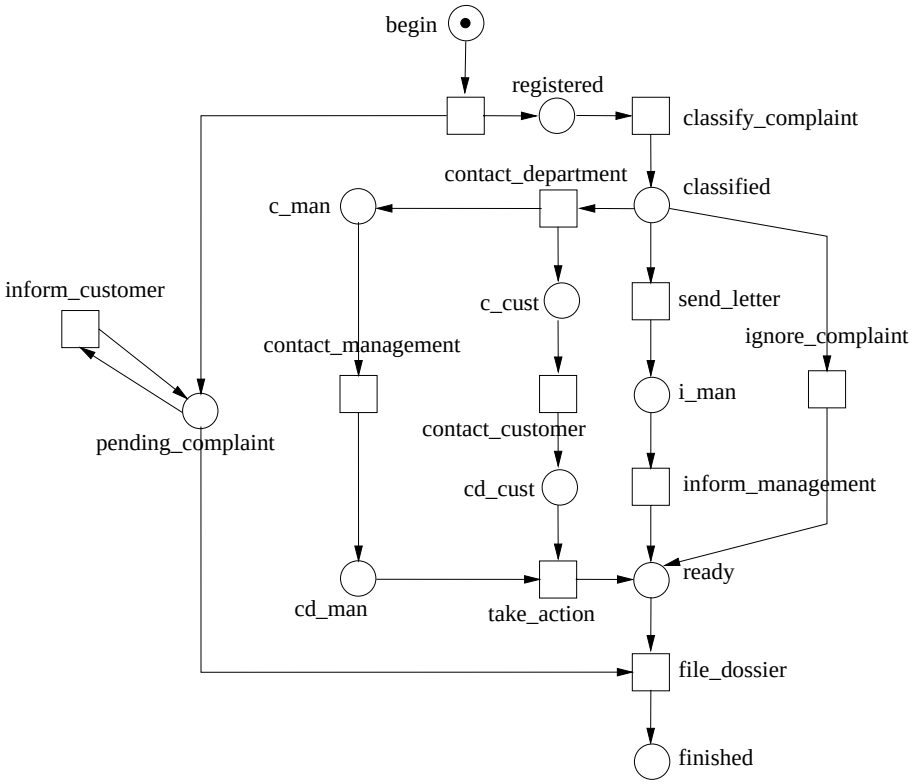

Figure 1.7 A Petri net describing the workflow procedure used to process complaints.

resource management in an integrated way by using an high-level Petri net extended with color and hierarchy. For more information the reader is referred to [AvHH95, AH96, AH96, AH95].

Experiences in the Sagitta 2000 project showed that Petri-nets can be used to model workflows in a natural manner. People that had no prior experience in computer science were able to specify workflow procedures. Although Petri nets are easy to use because of their graphical nature, they are well-founded and formal semantics are available. The fact that workflow procedures are specified using a technique with formal semantics is vital to the success of workflow projects such as Sagitta 2000. The fact that Petri nets have formal semantics has a number of advantages.

- A workflow procedure specified in terms of a Petri net is *unambiguous*, i.e., the meaning of each construction is clear and there is no room for multiple interpretations. This way, it is possible to avoid interminable discussions about the precise meaning a workflow procedure specification.
- A Petri net description of a workflow can serve as a *contract* between subdepartments. The formal semantics can be used to resolve conflicts over the interpretation of common workflow procedures.
- The interpretation of a Petri-net-based workflow procedure is *tool independent*; it does not change when a new version of the WFMS is released.
- The formal semantics allow for *reasoning about properties* of a given workflow procedure. It is possible to prove (the absence of) dynamic properties such as deadlock, livelock, etc.
- The formal semantics form a prerequisite for the application of all kinds of *analysis techniques*.

Many of today's available WFMSs provide ah-hoc constructs to model workflow procedures without any formal semantics. Moreover, there are WFMSs that impose serious restrictions on the workflow primitives shown in Figure 1.6. For example WANG's OPEN/workflow does not support the nesting of parallel flows. Some WFMSs also provide exotic constructs whose semantics is not 100% clear. To avoid these problems one could use a Petri-net-based WFMS having formal semantics. This does not mean that some 'syntactic sugaring' to facilitate the design process should be avoided. Note that the exchange of workflow process definitions between two Petri-net-based WFMSs is easy compared to the exchange of workflow process definitions between two WFMSs based on different concepts.

For more information on workflow modeling with Petri nets, the reader is referred to [EN93, Aal96, Aal97, AH97].

1.4 REASON 2: STATE-BASED INSTEAD OF EVENT-BASED

In contrast with many other process modeling techniques, the state of a case can be modeled explicitly in a Petri net. Process modeling techniques ranging from informal techniques such as dataflow diagrams to formal techniques such as process algebra's are *event-based*, i.e., transitions are modeled explicitly and the states between subsequent transitions are modeled implicitly. Today's WFMSs are typically event-based, i.e., tasks are modeled explicitly and states between subsequent tasks are suppressed. Figure 1.8 shows a typical diagram which defines a workflow procedure. The tasks *A, B, C, D, E, F, G* and *H* are represented explicitly in contrast to the state.

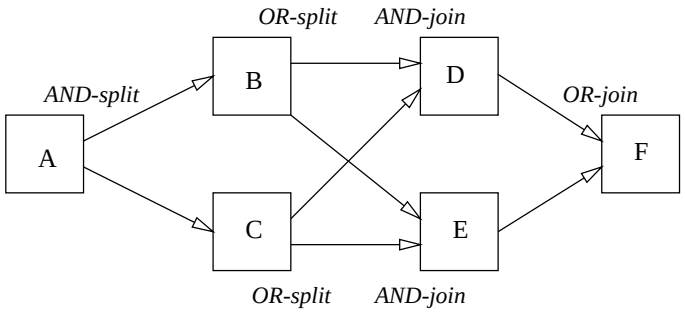

Figure 1.8 An event-based description of a workflow procedure.

If we convert the event-based description shown in Figure 1.8 to a Petri net, we obtain the net shown in Figure 1.9. The tasks are modeled by transitions and intermediate states are modeled by places. Note that in contrast to the description given in Figure 1.8 it is possible to refer to states between the execution of subsequent tasks.

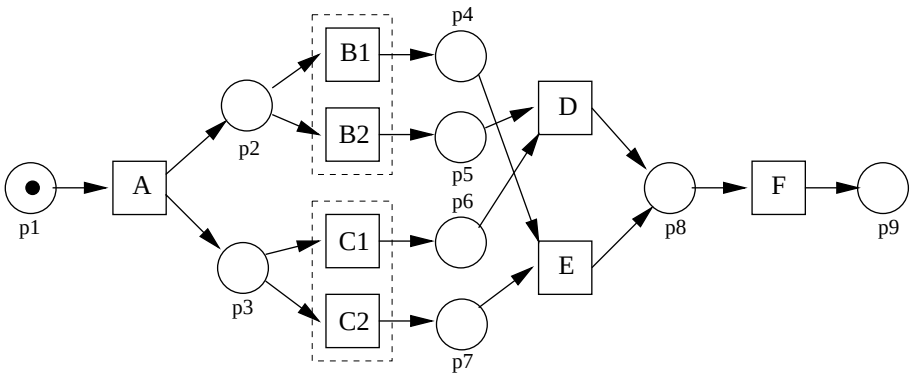

Figure 1.9 A state-based description of a workflow procedure.

The distinction between an event-based and a state-based[3] description seems to be very subtle, but turned out to be of the utmost importance in the Sagitta-2000 project. In general, there are several reasons for using a state-based description. These are discussed in the remainder of this section.

First of all, a state-based description allows for a clear distinction between the

enabling of a task and the execution of a task. Since the enabling of a task does not imply that the task will be executed immediately, it is important to have this distinction. To illustrate this, we need to discuss the *triggering* of tasks in more detail. The execution of a task instance for a specific case starts the moment the task instance is triggered. A task instance can only be triggered if the corresponding case is in a state which enables the execution of the task. Consider Figure 1.9. Task D can only be triggered for case c if there is a token in each of the input places of D which corresponds to c. There may be tasks which are not triggered by the WFMS itself. In Sagitta-2000 there are four kinds of triggering:

- Automatic: a task is triggered the moment it is enabled. This kind of triggering is used for tasks which are executed by an application which does not require human interaction.
- User: a task is triggered by a human participant, i.e., a user selects an enabled task instance to be executed. In a WFMS each user has a so-called 'in-basket'. This in-basket contains tasks instances that are enabled and may be executed by the user. By selecting a task instance the corresponding task instance is triggered.
- Message: an external event (i.e. a message) triggers an enabled task instance. Examples of messages are telephone-calls, fax messages, e-mails or EDI messages.
- Time: an enabled task instance is triggered by a clock, i.e., the task is executed at a predefined time. For example, the task 'remove document' is triggered if a case is trapped in a specific state for more than 15 hours.

Only for automatic tasks the enabling and the execution of a task coincide. Therefore, it is important model the intermediate states explicitly.

Another reason for the explicit modeling of states is the possibility of *competitive tasks*. Two tasks are competitive if they are both enabled and only one of them may be executed. Figure 1.11 shows two competitive tasks B and C. We use the symbols shown in Figure 1.10 to denote the way each task is triggered. Task A is triggered by an external message. The execution of task A is followed by the triggering of B or the triggering of C. If the user selects the instance of task B before some predefined time, then B is executed. Otherwise, task C is executed. Note that the execution of task C implies that task-instances have to be removed from the in-baskets of the participants which are allowed to execute task B. To model this situation we cannot use an event-based description. The choice to do task B or C is not made during the execution of task A. The choice

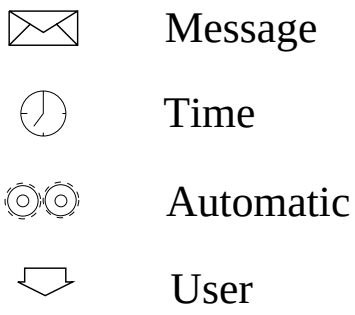

Figure 1.10 Four kinds of triggering.

to do B or C is implicitly made by the environment of the WFMS while the corresponding case marks place *p2*. There are many WFMSs which are unable to model the situation shown in Figure 1.11, simply because the intermediate state *p2* is suppressed. As a result an enabled task instance is required to be executed: once a task instance appears in an in-basket it remains there until it is executed.

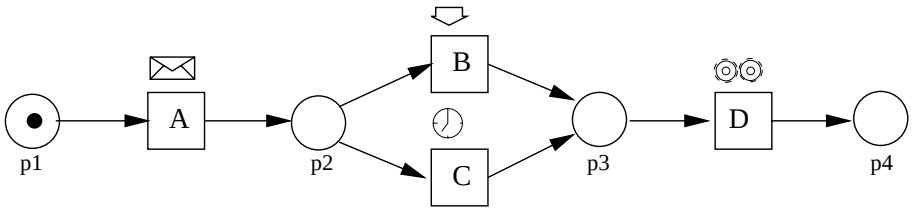

Figure 1.11 Task B and task C are competitive.

Sometimes it is necessary to withdraw a case. For many event-based WFMSs, this is difficult situation. Task instances have to be removed from the in-baskets of the participants. In a Petri-net based WFMS such a withdrawal is quite easy: simply remove all the tokens and triggers that correspond to the canceled case.

Today, a WFMS is often used within a single department. In the future, enterprise-wide workflow systems will become a reality. For example, Sagitta-2000 will be a distributed system composed of many independent autonomous workflow subsystems. Each of these subsystems runs on a local platform having

one server and many clients. Although each location is autonomous, cases are exchanged frequently. There are several reasons for moving a case from one location to another. There may be a compelling reason for such a transfer, e.g. a task can not be executed at the current location. However, a case transfer can also be issued to balance the workload. Anyhow, there has to be a way to transfer a case from one WFMS to another WFMS. For state-based WFMSs this is quite easy: remove all tokens which correspond to the case to be transferred and move them to the other WFMS. Note that this is only possible if the workflow procedures in the two WFMSs are compatible. Clearly, exchanging cases between event-based WFMSs is much more difficult.

There are many reasons for using a state-based WFMS instead of an event-based WFMS. Event-based WFMSs can only be used satisfactorily in situations where the workflow engine is leading, i.e., tasks are triggered by the WFMS instead of the environment of the WFMS. In many situations this is not very realistic. The WFMS should follow and guide the environment instead of imposing all kinds of restrictions.

1.5 REASON 3: ABUNDANCE OF ANALYSIS TECHNIQUES

Petri nets are marked by the availability of many analysis techniques. Clearly, this is a great asset in favor of a Petri-net-based WFMS. We have showed that the Petri net formalism allows for a representation of the workflow which is close to business process at hand, i.e., it is possible to model the workflow in a natural manner. This representation can be used as a starting point for various kinds of analysis. In a sense, the Petri net representation serves as an interface between the business process at hand and the method(s) of analysis. In fact, Petri nets provide a 'solver-independent' medium that can be used to make a concise 'blue-print' of the workflow definition we want to analyze. This blue-print may be used at different levels of decision making and can be used as a starting point for various means of analysis. Compared to the usual algorithmic approaches (where the emphasis is on the analysis process rather than the modeling process), this approach is characterized by the fact that during the modeling process the user is not shackled by the techniques which are going to be used to analyze the model.

For an overview of the many analysis techniques developed for Petri nets the reader is referred to [Hee94, Jen92, Mur89, SV90]. In general these techniques can be used to prove properties (safety properties, invariance properties, deadlock, etc.) and to calculate performance measures (response times, waiting

times, occupation rates, etc.). In this way it is possible to evaluate alternative workflows.

Let us focus on analysis techniques that can be used to prove properties of a given workflow procedure. By constructing the occurrence graph, we are able to verify whether a desired property holds. For example, we can use the occurrence graph to detect deadlocks and undesirable states. However, it is also possible to use techniques which exploit the structure of the underlying Petri net. For example, we can generate place invariants to verify safety properties. We also developed an analysis technique which verifies in polynomial time whether the workflow procedure satisfies the following requirements [Aal97]:

- There are no 'dangling tasks', i.e., tasks which do not contribute to the processing of cases.
- For any case, the procedure will terminate eventually. (Given some fairness assumption.)
- The moment the procedure terminates for a specific case, all references to this case have been removed.

A procedure which satisfies these requirements is called a *sound* workflow procedure. In [Aal97] this soundness property is defined formally and a technique is presented to verify this property in polynomial time. This technique is based on the rich theory developed for free-choice Petri nets [Bes87, DE95]. If we analyze the procedure shown in Figure 1.9 using this technique, we detect an error. The workflow procedure is not sound: executing task *A* for a specific case followed by *B1* and *C1* results in a deadlock. For the workflow procedure shown in Figure 1.9 this result is trivial. However, for the workflow procedures used by Sagitta-2000 it is far from trivial to verify the soundness property. (A workflow procedure contains typically 50 tasks.) Nevertheless, we succeeded in proving the soundness property for each of the procedures by using this technique.

Before introducing a new or revised workflow procedure it is important to have estimates of the important performance measures such as response times, waiting times and occupation rates of resources. Some of the leading WFMSs provide a simulation facility to evaluate the performance of a given workflow process without actually enacting the workflow procedure. There are many Petri-net based simulation tools. Therefore, it is easy to link a Petri-net-based WFMS to an existing simulation tool. If the duration of a task can be modeled by a negative-exponential distribution, then the corresponding Generalized

Stochastic Petri Net (GSPN) can also be analyzed by Markovian analysis techniques [MBC86]. If the duration of a task can be modeled by a pessimistic and an optimistic estimate (i.e. an interval), then the corresponding Interval Timed Colored Petri Net (ITCPN), can be analyzed using the MTSRT method presented in [Aal93]. In either case, standard tools are available for performance analysis of the workflow process at hand.

Clearly, the abundance of analysis techniques developed for Petri nets, enables the user of a Petri-net-based WFMS to analyze a workflow process in various ways (including simulation). In the Sagitta-2000 project we used the Petri-net-based analysis tools ExSpect [ASP94] and INA [Sta92] for simulation purposes and structural analysis.

1.6 CONCLUSION

Today's situation with respect to workflow management software is comparable to the situation as regards to database management software in the early 70-ties. In the beginning of the 70-ties most of the pioneers in the field of DBMSs were using their own ad-hoc concepts. This situation of disorder and lack of consensus resulted in an incomprehensive set of DBMSs. However, emerging standards such as the Relational Data Model [Cod70] and the Entity-Relationship Model [Che76] led to a common formal basis for many DBMSs. As a result the use of these DBMS boosted. There are many similarities between today's WFMSs and the DBMSs of the early 70-ties. Despite the efforts of the Workflow Management Coalition a real conceptual standard is missing. As a result many organizations are reluctant to use existing workflow management software. In our opinion Petri nets constitute a good basis for standardization. Inspired by practical experiences, we have come to realize that many of the features of the Petri net formalism are useful in the context of workflow management. In this paper we have given three solid reasons for using a Petri-net-based WFMS. For the Sagitta-2000 project these reasons turned out to be crucial:

- Formal semantics despite the graphical nature. For the Sagitta-2000 team it was important to have a concise set of terms and an unambiguous diagraming technique. The diagraming technique is easy to use for the people involved in the Sagitta-2000 project. Moreover, the formal semantics of the diagraming technique enable the use of diagrams as a 'contract' between the (sub)departments cooperating in this project.

- State-based instead of event-based. In the beginning the Sagitta-2000 was hampered by the use of event-based diagrams typically used in many workflow products. It was difficult to handle case transfers, rerouting

of cases, case withdrawals and external triggers. By using a state-based approach these problems were solved quite easily. This was one of the prime reasons for adopting Petri nets.

- Abundance of analysis techniques. Simulation was used to validate the business processes and new concepts. In addition advanced Petri-net-based analysis techniques were used to verify the correctness of the complex workflow procedures for Sagitta-2000. These techniques allow for the verification of future changes of the workflow procedures.

Based on these reasons the Dutch Customs Department decided to select COSA for the local platform and to build a Petri-net-based workflow controller for the central platform.

Acknowledgments

The author would like to thank the Sagitta-2000 team, in particular Peter van der Toorn, Silvia de Kloe, Jaap Rigter and Hans-Rob de Reus, for their contributions to the results reported in this paper.

Notes

1. The terms related to the processing of Dutch Customs declarations are too specialized to be translated. This is the reason we focus on the lessons learned rather than expatiating on on specific Customs processes.
2. We use a fictitious example because the workflow procedures in Sagitta 2000 are too complex and require extensive knowledge of the jargon and the processing of Customs declarations.
3. We use the term state-based to denote that states are modeled explicitly. Clearly, a state-based description also incorporates state transitions, i.e., events.

References

[Aal93] W.M.P. van der Aalst. Interval Timed Coloured Petri Nets and their Analysis. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 453–472. Springer-Verlag, Berlin, 1993.

[Aal96] W.M.P. van der Aalst. Petri-net-based Workflow Management Software. In A. Sheth, editor, *Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems*, pages 114–118, Athens, Georgia, May 1996.

[Aal97] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azema and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, Lecture Notes in Computer Science, page (to appear). Springer-Verlag, Berlin, 1997.

[AH95] W.M.P. van der Aalst and K.M. van Hee. Framework for Business Process Redesign. In J.R. Callahan, editor, *Proceedings of the Fourth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 95)*, pages 36–45, Berkeley Springs, April 1995. IEEE Computer Society Press.

[AH96] W.M.P. van der Aalst and K.M. van Hee. Business Process Redesign: A Petri-net-based approach. *Computers in Industry*, 29(1-2):15–26, 1996.

[AH97] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Modellen, Methoden en Systemen (in Dutch)*. Academic Service, Schoonhoven, 1997.

[ASP94] ASPT. *ExSpect 4.2 User Manual*. Eindhoven University of Technology, Eindhoven, 1994.

[AvHH95] W.M.P. van der Aalst, K.M. van Hee, and G.J. Houben. Modelleren en Analyseren van Workflow: een Aanpak op Basis van Petri-netten. *Informatie*, 37(11):590–599 (in Dutch), 1995.

[Bes87] E. Best. Structure theory of Petri nets: the free choice hiatus. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*, volume 254 of *Lecture Notes in Computer Science*, pages 168–206. Springer-Verlag, Berlin, 1987.

[Che76] P.P. Chen. The Entity-Relationship Model: Towards a unified view of Data. *ACM Transactions on Database Systems*, 1:9–36, Jan 1976.

[Cod70] E.F. Codd. A Relational Model for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[DE95] J. Desel and J. Esparza. *Free choice Petri nets*, volume 40 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, Cambridge, 1995.

[EN93] C.A. Ellis and G.J. Nutt. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *Application and Theory of*

Petri Nets 1993, volume 691 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1993.

[Hee94] K.M. van Hee. *Information System Engineering: a Formal Approach*. Cambridge University Press, 1994.

[HL91] K. Hayes and K. Lavery. *Workflow management software: the business opportunity*. Ovum, 1991.

[Jen92] K. Jensen. *Coloured Petri Nets. Basic concepts, analysis methods and practical use.* EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1992.

[Kou95] T.M. Koulopoulos. *The Workflow Imperative*. Van Nostrand Reinhold, New York, 1995.

[MBC86] M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems*. The MIT Press, Cambridge, 1986.

[Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[Rei85] W. Reisig. *Petri nets: an introduction*, volume 4 of *Monographs in theoretical computer science : an EATCS series*. Springer-Verlag, Berlin, 1985.

[Sch96] T. Schäl. *Workflow Management for Process Organisations*, volume 1096 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1996.

[SL96] Software-Ley. *COSA User Manual*. Software-Ley GmbH, Pullheim, 1996.

[Sta92] P.H. Starke. *INA: Integrierter Netz Analysator, Handbuch*, 1992.

[SV90] M. Silva and R. Valette. Petri Nets and Flexible Manufacturing. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pages 274–417. Springer-Verlag, Berlin, 1990.

[WFM96] WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.