KATHOLIEKE
UNIVERSITEIT
LEUVEN

DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

RESEARCH REPORT 0009

AN ARCHITECTURE FOR BRIDGING OO AND BUSINESS PROCESS MODELLING

by
M. SNOECK
S. POELMANS
G. DEDENE

D/2000/2376/09

An architecture for bridging OO and Business Process Modelling

M. Snoeck - S. Poelmans -G. Dedene

submitted for TOOLS Europe 2000

Abstract. Workflow systems and object-oriented (OO) technology have undoubtedly been some of the most important domains of interest of information technology over the past decade. Both domains however, have largely evolved independently, and not much research can be found in which OO principles and concepts have been applied to workflow systems or vice versa. In this paper we show how the two domains can be integrated. By integrating both domains, business process modelling can benefit from the advantages of the object-oriented approach. On the other hand, a more process oriented approach to object-oriented development would enhance the organisational fit in of object-oriented information systems development. The architecture that results from this integration is a tier-based one with a separate tier for workflow aspects.

Keywords: object-oriented analysis, business process modelling, architecture

1. INTRODUCTION

Workflow systems and object-oriented (OO) technology have undoubtedly been some of the most important domains of interest of information technology over the past decade. Both domains however, have largely evolved independently, and not much research can be found in which OO principles and concepts have been applied to workflow systems or vice versa. In fact, both domains are complementary: what is missing in one domain is present in the other and vice versa. The next two paragraphs briefly sketch the situation of both domains. The third paragraph lists some of the advantages that can be gained by integration.

1.1 The lack of a functional view in Process Modelling

A workflow system can be defined as information technology that can be used to model, enact and adapt business processes [6]. A business process consists of a number of activities that have to be executed in some order by several end-users in order to fulfil the business goal. In the modelling phase, the activities, their order of execution and the agents that are responsible for the activities are determined. One of the principal issues of workflow systems is their ability to cope with changing requirements inside and outside an organisation.

Workflow (management) systems have been developed to design, execute, control and adapt business processes in an organisational context. Workflow systems originate from office automation developments in the 80's. Whereas typical office automation applications (like text editors, spreadsheets and databases) are designed to support individual end-users, workflow systems are created to support a process view on business activities. Not only individual users with individual activities but also an entire process with several interdependent users and activities directed towards one common business goal should be supported. In this way, costs can be diminished and/or (customer) services can be improved so that workflow systems can be used as a strategic advantage.

In the literature on workflow modelling, several techniques are proposed to define and represent the structure of a business process (petri-nets, flow charts, etc.). In most cases, the method to be followed is imposed by the vendor of the workflow package [10]. The philosophy and theoretical approach that are implicitly or explicitly applied in a particular workflow system, often leaves the designer with no choice. The Action Workflow System for example is built according to the principles of the speech act theory and forces the developer to use the specific diagrams and representations of this theory [13]. Another example is the Trigger Model of Joosten [9], that uses petri-nets to model activities.

Nevertheless, some general requirements can be put forward that are necessary to be able to model a process:

1. Processes, activities and operations need to be defined in a hierarchical manner:

A process is the highest level of the hierarchy. Process design typically requires a top down decomposition of high level processes into sub-processes down to atomic operations. More precisely, a process might be composed of procedures, which are defined as a limited sequence of activities. Each activity (also called a process step) can be further subdivided into sub-activities or subtasks. Notice that it is the division of labour in the organisation that determines the subdivision in activities and sub-activities. Activities and tasks might have a different meaning in different organisational theories. In the field of workflow modelling however, both terms are often used interchangeably and we will do the same in this paper.

2. The sequence of procedures, activities and sub-activities is crucial:

The main goal of a workflow system is in fact the automation or support of the co-ordination between activities and between activities and agents. Co-ordination can be defined as the management of dependencies [12]. In what way does one activity depend on the results of another activity? The modelling of dependencies constitutes the heart of workflow modelling. The existence of dependencies implies a certain order of execution. Some (sub)tasks cannot be performed before previous tasks have been completed; other tasks need to be executed in parallel, and so on.

3. Agents are humans or computer applications:

Not only the dependencies of activities need to be modelled, but also the interaction between activities and actors (from now on called "agents") needs to be planned ahead. An agent is able to perform different activities, and different agents can perform a specific activity. Agents can be human end-users or computer applications that perform activities. When human agents execute certain tasks, they might be assisted by computer applications to

support them. Only when applications are directly coupled to the workflow system, they are considered as agents. When an application is evoked by the workflow system and when the application performs a certain activity without any intervention of the end-user, it is called an autonomous agent. An agent is called semi-autonomous when it is directly coupled to a workflow system, although an intervention of the end-user is still required.

4. Agents are assigned to roles:

Agents are assigned to activities via the construction of roles. A role defines the responsibility for the performance of a (collection of) task(s). In general, the responsibility for one activity should not be assigned to more than one role. One role may be responsible for several activities, but one activity should belong to one role [11].

5. The specification of the business process (or workflow) needs to be a persistent artefact:

The workflow process is specified as a model in a formal textual and/or visual language. This model specification or definition is used whenever a new workflow instance needs to be created. Each time a workflow instance is created, the persistent workflow model is needed for controlling, supervising and recording the performed activities. Moreover, in order to monitor and improve performance, it is often also required to save the states of instantiated processes that have been enacted. Historical data regarding the actual course of processes can be useful and even necessary to improve the persistent process model.

The dependencies between activities and between activities and agents can be considered as the control logic of business processes. The functional part contains the necessary data and the applications that (partly) perform the activities (the non-human agents). The isolation of the control structure from the data and functional structure is a typical characteristic of workflow

systems [21]. The process logic is modelled but the functional part is not taken into consideration. In this way, alterations in the progress of the work can be represented in the workflow system by simply adapting the parameters of the process logic. Since the functional part is not considered, it is not clear however how the functional part will be affected by a change in the control logic. A simple change in the process might have considerable consequences for the functional part [15].

In sum, when modelling a business process (or workflow), an activity-view is advocated. Activities and roles are defined (either or not on a detailed level of granularity) and coupled in a global process. Hence, a business process is divided into a function-oriented part - the activities - and a process-oriented part - the relationship between activities. In a workflow system, the function-oriented part is supposed to be given, whereas the process-oriented part (the process logic) is modelled and supported [16].

1.2 The lack of a process view in OO development

In object-oriented development, the situation is just the opposite: the primary emphasis is on the specification and development of the functional part, whereas the process part is largely neglected or supposed to be given [e.g. 1, 2, 4, 5, 7, 8, 14, 19].

The dynamic behaviour and interaction between object (classes) is modelled in OO development systems by means of several representations like event charts, state transition diagrams, sequence diagrams, etc. Whereas a state transition diagram only models the behaviour of one class, other diagrams (like the sequence or collaboration diagram) model the interaction of several classes. In this way, process logic is implicitly represented and a process-oriented view is in fact not completely omitted. However, some process modelling requirements that we proposed in the previous paragraph are not met. In the first place, the top down decomposition of processes (requirement 1) is barely supported in object-oriented development. Functional decomposition, a vital concept from the structured programming world, is

often considered as old-fashioned and uneffective by object-oriented developers [22]. In addition agents are not assigned to activities (requirement 3 and 4); and process logic is not designed to be implemented as an (persistent) application (requirement 6).

One way to introduce some of the business process aspects information systems analysis is the use of Use Cases [8, 2]. Use cases describe *what* the system is supposed to do from the actor's perspective. They do not describe *how* the system should be designed and implemented. In fact, the use cases serve as input for the construction of formalised and structured object classes. The object classes should provide the functionality to support the use cases. The concept of actor in use cases does however not completely match with the concept of agent in workflow modelling. Moreover, use cases are not intended to model the assignment of agents to activities and the co-ordination between activities.

Next to use cases, other dynamic representations (like state transition diagrams and sequence diagrams) are created in the development phase. These diagrams are built on the basis of one or more object classes and they represent processes or procedures. The process logic in this type of diagrams is however mainly relevant for the functional aspects of the application. In some cases, aspects of a business process can be found in this type of diagrams. However, such business process logic is not explicitly and separately implemented. It is embedded in the classes and changes in business rules need to be done on object(class) level. In this way, an adaptation of object classes is necessary to change the business process logic.

1.3 Advantages that can be gained by integration

Advantages for workflow systems

Workflow systems have been designed to capture and optimise business processes. However, state-of-the-art workflow systems are mostly not object-

oriented and they have intrinsic disadvantages that can be solved by applying the object-oriented methodology.

A first disadvantage has to do with the separation of the process-oriented part from the function-oriented part. Computer agents that support certain activities are regarded as a monolithic black boxes. This means that adaptations in the process logic might have considerable consequences for the supporting applications. As a result, adaptability is not guaranteed. Because of its modularity and encapsulation, the OO approach is well suited to solve this problem. Required changes in the functional part can be limited to the object classes involved, without jeopardising the consistency of the entire system.

Schreyjak [16] points to another disadvantage of workflow systems. Workflow systems are often used in a heterogeneous infrastructure. Therefore, they need to be platform independent. However, this places a burden on the supporting applications that need to support the activities. When applications are black boxes, it is not sure how well they are suited to be ported to heterogeneous environments. Therefore, Schreyjak advocates the use of components to build autonomous agents. Components can be off-the-self of newly built. Using the advantages of the object-oriented approach and standards like CORBA, their portability can more easily be guaranteed.

Applying the OO approach should also be considered when modelling the process-oriented part. An OO workflow model can claim the same advantages of OO information systems in general. Using the object-oriented method results in workflow object classes (representing the process logic) that can be reused, ported (for example by making them compliant with international standards) and adapted.

Advantages for OO development

A separation of concerns is a key elements in keeping systems maintainable and adaptable. In current object-oriented system development practice, the organisational aspect of an information system is often not explicitly modelled. And when it is, it is not always taken as important element in guiding design decisions. By integrating business modelling concepts into

object-oriented modelling, the link between the services that an information system has to render and the organisational elements becomes more apparent. This can be an important help in designing more adaptable systems. In addition, when workflow elements are not modelled separately, they are often hidden in the procedural logic of class-methods. The explicit separation of workflow elements from process elements that are inherent to the domain or to the procedural logic of an implementation also allows for more adaptable systems. For example, sequence constraints on events that result from the business logic are part of the domain model (e.g. in a library, the return of a copy to the library must be preceded by a borrowing event). These type of sequence constraints are less likely to change over time than sequence constraints that are the result of workflow aspects (e.g. if a member of the library does not show up after five reminders, set all the books (s)he borrowed to the state "lost").

2. THE MISSING LINK: BUSINESS EVENTS

A full-fledged information systems development method should take all aspects into account: aspects of business process modelling and aspects of the functional part should be linked together. In the architecture proposed below, business events are used as hinge concept between business process modelling concepts and object-oriented modelling concepts.

Business process modelling takes an action- and process-oriented view on the domain. As a result , task and activities are easier to formulate in terms of business events than in terms of business objects (which are better for modelling to structural aspects). From a business modelling perspective, only business events are of particular interest. Information system events such as keyboard actions and mouse clicks are modelled as elements in the information system, but are not relevant elements in a business process model.

Also in object-oriented modelling, business events have an essential role to play: they appear as triggers for the execution of object methods. Some object-oriented development methods go even one step further and identify

business events as a fundamental component of an object-oriented real-world model. This is, for example, the case for Syntropy [4], OO-SSADM [14] and MERODE [18, 19].

In fact, events are a fundamental part of the structure of experience [4]. Events are atomic units of action: they represent things that happen in the world. Without events nothing would happen: they are the way information and objects come into existence (creating events), the way information and objects are modified (modifying events) and disappear from our universe of discourse (ending events). Events are not objects. However, we might choose to record the fact that an event has happened by recording the occurrence of this event as an object. For example in a banking environment, "*withdraw money*" is an event that modifies the state an object "BANK ACCOUNT". We can keep track of all withdrawals by defining "WITHDRAWAL" as an additional object type. An event *withdraw* will from then on have a double effect: it will *modify* the state of an account and *create* a withdrawal. During the analysis stage, it would be irrelevant to determine how both objects will be notified from the occurrence of the withdrawal event. We therefore assume, just as in Syntropy [4], OO-SSADM [14] and MERODE [18, 19], that events are broadcasted.

The separation of business events (also called real-world events) from information-system events allows a more user-oriented and task-oriented view of information system design. Business events are those events that occur in the real-world, even if there is no information system around. Information-system events are directly related to the presence of a computerised system. They are designed to allow the external user to register the occurrence of or invoke a real-world event. For example, the use of an ATM-machine to withdraw money from one's account will invoke the business event "*withdraw*" by means of several information-system events such as "*insert-card*", "*enter PIN-code*", "*enter amount*", and so on. Once events have been identified in the domain model, the whole domain model can be considered as one component, which interface is the set of all events that allow to create, modify and update the information contained in the domain model. User functions (or information system services) are then nothing more than a way to

invoke these business events. The user function will translate information-system events such as mouse clicks and keystroke actions into the invocation of one or more domain model event.

The figures below represent a meta-model for the concepts used in this system development approach. In the proposed architecture, business events are used as the bridging concept between workflow activities and information system design. In a first step, business processes are modelled at a conceptual level by decomposing them down to the activity level and by indicating which business event each activity invokes. The domain is modelled at the conceptual level by identifying domain object classes and by indicating by which business event they are affected. The effect of an event on a domain object class is recorded in a domain object class method. Fig. 1 shows a meta-model relating the modelling concepts at this stage of the specification process. By using business events as linking concept, business process modelling and domain modelling can be performed at conceptual level. Notice that we assume here that workflow concepts are modelled in an object oriented way, such as for example in the $TriGS_{flow}$ model [11].

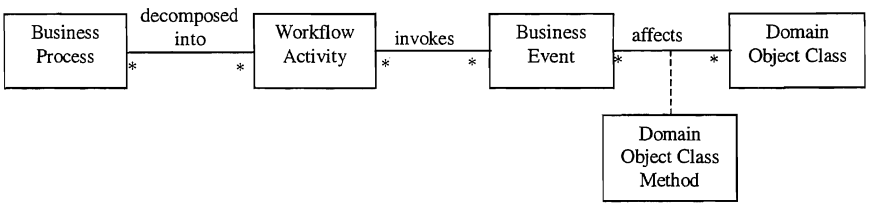

Fig. 1. Meta-model for conceptual modelling

In a second step the business processes and the business domain are analysed in search for information support. So, next to the description of the domain of interest in the domain model, we need a specification of the services (also called user functions) that the information system has to render to the prospective users. This part of the specification is closely related to the specification of the workflow model: it is the description of the functional support for the tasks of the workflow model. The activities that have to be

performed by agents can be further classified as manual, interactive or fully automated. Interactive and automated activities are realised by means of an information system service. These information systems services interact with the domain model by invoking business events. Fig. 2. represents the meta model for this more detailed level of specifications.

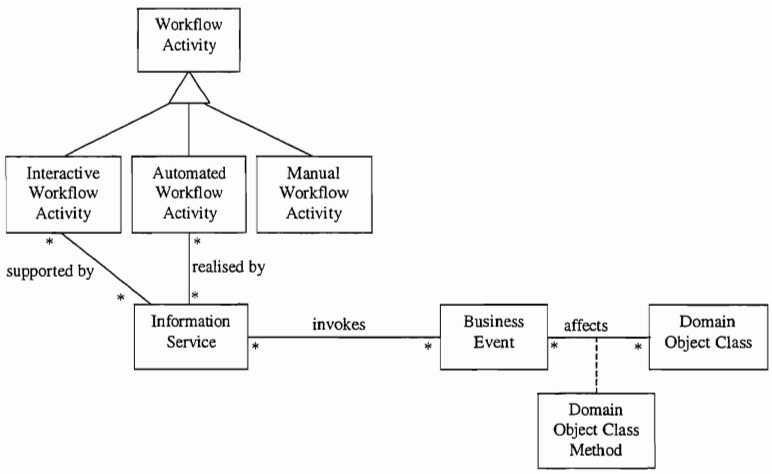

Fig. 2. Information systems modelling meta-model.

3. A CASE-STUDY

In the next paragraphs, the ideas are explained and illustrated by means of a real-life case-study taken from the university administration. The domain that is covered in this case -study is the acquisition of materials by the university and the payment of bills. First a business process model is developed. Next, an object-oriented real-world model is developed. In the third paragraph we move into a more detailed analysis. The required information system services are identified and linked to the tasks identified in the business process model.

3.1 The Business Process Model

The main stakeholders in the acquisition and financial processes are the local university units (such as faculties, departments, research groups, labs, ...), the central financial unit, the central acquisition unit and the suppliers.

When a local unit wishes to acquire some materials, an order has to be created using the information system (rather than a word-processing tool). The order is then printed, signed and sent to the supplier. The supplier delivers the goods and sends the bill to the central financial unit of the university. The bill is scanned and registered in the system. A notification is sent to the local unit that placed the order. There the electronic image of the bill can be viewed to check the bill. If the bill is approved, the central financial unit is notified of the approval so as to be able to pay the bill. When something is wrong with the bill or the delivered goods, step 5 is followed by a letter to the supplier and the business process is resumed at step 3. The basic business process is shown in Fig. 3 In this figure, full arrows represent a paper-based communication and dashed arrows represent an electronic communication.

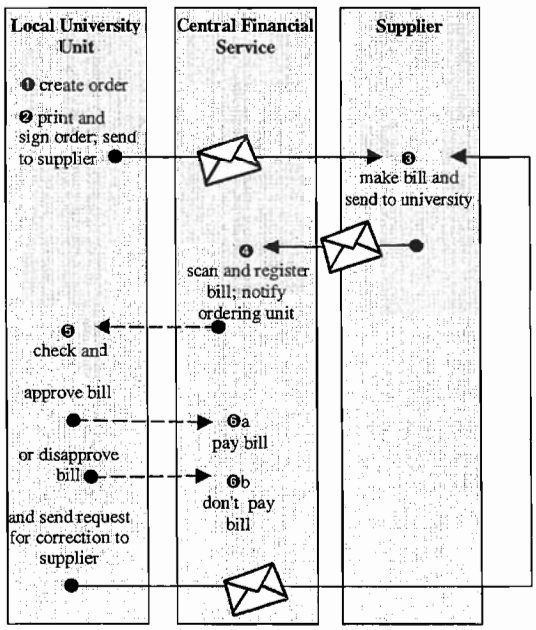

Fig. 3. Acquisition Business Process

This business process is the "standard" way of working on which a lot of variants exist. The first variant deals with acquisitions without order. For

example, a researcher goes to a book store, buys a book, and does not pay the book but instead requests a bill which has to be paid by the department. In this case, the local unit will create a "request for payment". In the information system, this type of document contains the same information as an order. After the request for payment has been created, the local unit sends the bill to the central financial unit. Because the bill refers to a request for payment, it has not to be approved any more and is thus directly paid. This business process is shown in Fig. 4.

Sometimes however, the goods are paid directly by the purchaser (usually a member of the personnel), and refund is requested afterwards. In that case, from the point of view of the university, the supplier is the member of the personnel that becomes the supplier of the goods, because it is this person that has to be paid. Rather than making a bill, the person to be refunded has to fill in a form called "refund request". The rest of the process is similar to the direct acquisition process (see Fig. 5).

A third variant on this business process is the acquisition of materials that falls under the regulation of public contracts. If the amount of the acquisition exceeds a certain limit, the university has to invite at least three possible suppliers for a tender. This task is taken care of by the central acquisition unit. The local unit now creates a "request for order". This request is handled by the Central acquisition unit. Once agreement is reached on the supplier that will receive the order, the central acquisition unit converts the request for order into an order and the basic business process is followed (Fig. 6).

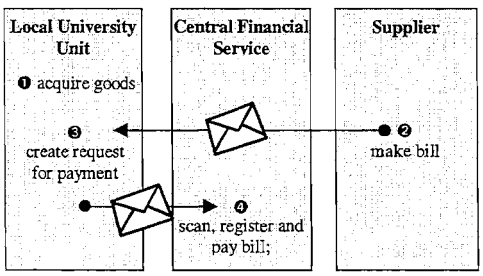

Fig. 4. Direct acquisition

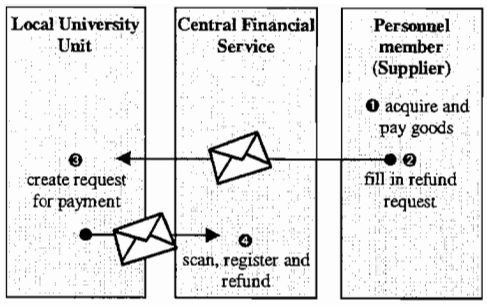

Fig. 5. Direct prepaid acquisition

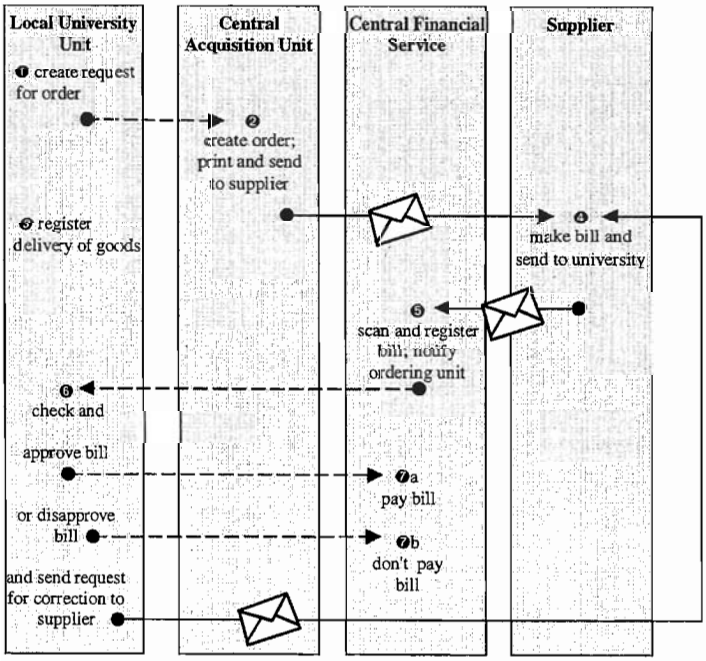

Fig. 6. Acquisition under the regulation of public contracts

3.2 The domain model

The domain model specifies business rules in terms of objects, associations between objects and business events. The structural part of the domain model for the acquisition and financial administration domain is given in Fig. 7. Notice that a bill refers to exactly one payment document. This means that whenever the central financial service receives a bill, they must be able to link it to an existing payment document to be able to register it in the system.

The behavioural part of the domain is modelled in terms of business events and the way these events affect the domain objects. This can be modelled by means of an object-event table as shown in Fig. 8. The business process descriptions are an obvious input for finding relevant business event types.

In the object-event table, there is one column for each business object and one row for each business event. A row-column intersection is marked with a 'C' when the event creates the object, with a, 'M' when it modifies the state of the object and with an 'E' when it ends the life of the object. A marked entry in a column means that the object class has to be equipped with a method to implement the effect of the event on the object. For example, the 'M' in the fifth column on the row labeled 'register_delivery' means that the event register_delivery modifies the state of an order. The class ORDER will thus be equipped with a method register_delivery that allows to record the change of state and its associated effect on the object. The table in Fig. 8 shows a minimal set of marked entries. For example the event cr_orderline is only marked as creating event for ORDER LINE. It could also be marked as modifying event for ORDER as adding an order line modifies the total price of the order. The discussion of which entries to mark or not to mark is beyond the scope of this paper but can be found in [18, 19].

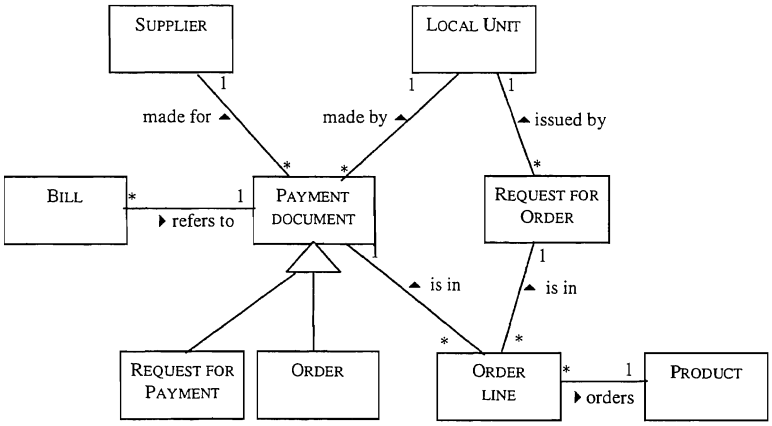

Fig. 7. Object-relationship diagram for the acquisition and financial administration.

Additionally, ordering on business events can be specified. These can originate from two sources: either form business rules (e.g. it is impossible to modify an order that was not created before) or from a work-organisational point of view (it is not allowed to pay a bill for which no payment document was created). In the first case, the sequence constraints are modelled as lifecycles attached to object classes. These can for example be specified by means of state machines. The second type of sequence constraints are modelled as part of the workflow specifications.

	Local Unit	Supplier	Payment document			Order line	Product	Bill	Request_ for_Order
				Order	Payment request				
cr_local_unit	C								
end_local_unit	E								
cr_supplier		C							
end_supplier		E							
cr_payment_ document			C						
end_payment_document			E						
cr_order				C					
end_order				E					
cr_payment_ request					C				
end_payment_ request					E				
register_delivery				M					
cr_orderline						C			
end_orderline						M			
mod_orderline						E			
cr_poduct							C		
end_product							E		
cr_bill								C	
mod_bill								M	
approve_bill								M	
reject_bill								M	
end_bill								E	
pay_bill								M	
cr_request_for order									C
Convert_request_ to_order				C					E

Fig. 8. Object-event table with basic participations

3.3 The service model

Next to the description of the domain of interest in the domain model, we need a specification of the services (also called user functions) that the information system has to render to the prospective users. This part of the specification is closely related to the specification of the workflow model: it is the description of the functional support for the tasks of the workflow model. From the business processes described in section I, we can derive a list of services that must be present in the information system. Table 1 lists the services that are required by the local-unit staff and Table 2 lists those required for the staff of the Central Financial Unit. A similar table can be made for the Central Acquisition Unit. Each table identifies the basic activities that require functional support for the information system. The link

with the domain model is established by identifying which business event is invoked by the service.

The specification of the services can then further be refined by the specification of the dialogue structure by means of regular expressions [3, 20] or state charts [2], by the specification of user interface part, and by the identification of the required objects and their collaborations [17].

Table 1: Services for Local Unit staff

Services for Local Unit	Business Process	Task	Invoked business event
Create Order	Basic Acquisition Process	❶	cr_order, cr_order_line, end_order_line, mod_order_line
Print Order	Basic Acquisition Process	❷	
Approve bill	Basic Acquisition Process	❺	approve
	Public contract	❻	
Notify central financial service of approval of bill	Basic Acquisition Process	❺	
	Public contract	❻	
Create Request for payment	Direct Acquisition Direct prepaid Acquisition	❶	cr_payment_request, cr_order_line, end_order_line, mod_order_line
Create Request for order	Public Contract	❶	cr_request_for_ order, cr_order_line, end_order_line, mod_order_line
Register delivery of goods	Public Contract	❸	register_delivery

Table 2: Services for Central financial Unit staff

Services for Central Financial Unit	Business Process	Task	Invoked business event
Register bill	Basic Acquisition Process	❹	cr_bill
	Direct Acquisition	❹	
	Direct Prepaid Acquisition	❹	
	Public Contract	❺	
Scan bill	Basic Acquisition Process	❹	mod_bill
	Direct Acquisition	❹	
	Direct Prepaid Acquisition	❹	
	Public Contract	❺	
Notify creator of order of pending approval of bill	Basic Acquisition Process	❹	
	Public Contract	❻	
register payment of bill	Basic Acquisition Process	❻	Pay_bill
	Direct Acquisition	❹	
	Direct Prepaid Acquisition	❹	
	Public Contract	❼	

4. CONCLUSION

In this paper we have proposed an architecture that integrates the concepts of object-oriented modelling with those of business process modelling. The integration leads to advantages for both fields of interest. The main advantages for workflow systems are a better adaptability for the functional part and the general advantages of the object-oriented approach such as e.g. portability across platforms. The main advantages for object-oriented development are a better organisational fit in and a better separation of concerns in the design of systems.

A closer look at the proposed meta-model reveals that the separation of concerns is already apparent in the structure of the meta-model (Fig. 9). The resulting architecture is a tier-based one, with a specific tier for business process modelling concepts. If we add the user interface tier, we obtain the full picture (Fig. 10). In the object-oriented community, a generally accepted architectural structure is not yet agreed upon. In general, a three-tier architecture is widely approved. The specific contents and meaning of the tiers can vary considerably however. Jacobson et al.[8] for instance, distinguishes three tiers: the domain tier (that is persistent), the control tier (with business rules and application logic) and the GUI tier (with only presentation logic). Fowler [7] proposes the data tier, the domain tier (with business logic) and the application logic tier (with specific application and presentation logic). The architecture that we propose has

- a domain tier, which contains persistent domain objects and the business domain rules. This tier can also be extended with controller classes such as an event handler and with DBMS classes.

- an information service tier which contains most application logic. This tier will mainly contain transient objects, although additional persistent object can be defined to support services. Commit and roll-back features and scheduling aspects are also defined at information service level.

- a User Interface tier, which contains the presentation logic and other interface aspects such as a first validation of user input.

- and an Workflow tier, which contains all business process logic and acts as a driver and controller for business procedures.

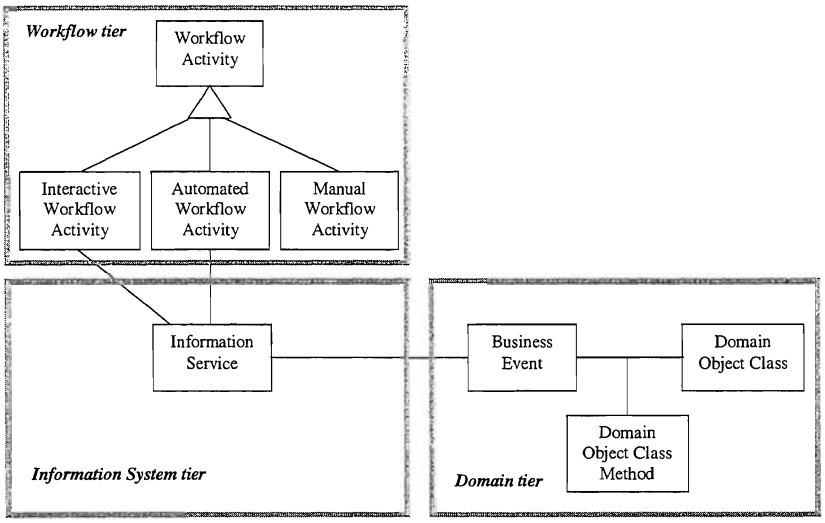

Fig. 9. Tiers in the integrated meta-model.

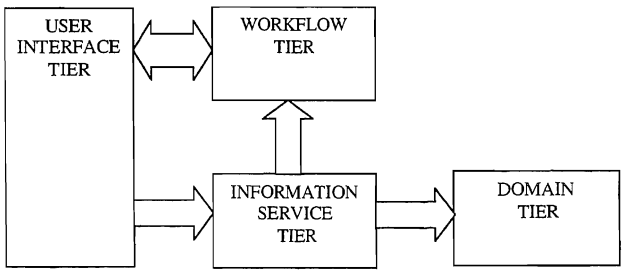

Fig. 10. A tier-based architecture with four tiers

For a better integration of workflow aspects in object-oriented systems development, we should at least consider the addition of a workflow tier to the classical three tier architectures. The paper has also demonstrated that business events can be used as hinging concept between the workflow tier and the information systems concepts.

5. REFERENCES

1. Booch, G., J. Rumbauch & I. Jacobson, The Unified Model Language User Guide, Addison-Wesley, 1999, 482 pp
2. Booch Grady, Rumbaugh James, Jacobson Ivar, *The unified modelling language user guide*, Addison Wesley, 1999
3. Coleman Derek et al, *Object-oriented development: The FUSION method*, Prentice Hall, 1994
4. Cook Steve, Daniels John, *Designing object systems: object-oriented modelling with Syntropy,* Prentice Hall, 1994
5. D'Souza, D.F. & A. C. Wills, Objects, Components and Frameworks with UML, The Catalysis Approach, Addison-Wesley, 1999, 785 pp..
6. Ellis C.A. & G.J. Nutt, Modelling and Enactment of Workflow Systems, Technical report, Department of Computer Science, University of Colorado, 1993.
7. Fowler, M., Analysis Patterns, Reusable Object Models, Addison Wesley Longman, 1997, 357 pp.
8. Jacobson, I., Christerson, M., Jonsson P. et al., *Object-Oriented Software Engineering, A use Case Driven Approach*, Addison Wesley, Rev. 4th pr., 1997.
9. Joosten, S. Trigger modelling for workflow analysis. In Proceedings of CON '94: Workflow Management, Challenges, Paradigms and Products, October 1994, München, pp. 236-247.
10. Joosten, S. Werkstromen : een overzicht, in Informatie, jaargang 37, nr. 9, pp. 519-528.
11. Kappel, G., P. Lang, S. Rausch-Schott, & W. Retschitzegger, Workflow management based on objects, rules and roles, In Bulletin of the Technical Committee on Data Engineering, March 1995,18(1), pp. 11-18.
12. Malone, T. W. & K. Crowston, The Interdisciplinary Study of Co-ordination, In ACM Computing Surveys, Vol. 26, No. 1, March 94, pp.87-119.
13. Medina-Mora, R., Winograd, T., Flores, & Flores, F. The Action workflow approach to workflow management technology. In Proceedings of the Conference on Computer Supported Co-operative Work '92, New York, Nov. 1992, pp. 281-288.
14. Robinson Keith, Berrisford Graham, Object-oriented SSADM, Prentice Hall, 1994
15. Schreyjak, S., Coupling of Workflow and Component-Oriented Systems, In Second International Workshop on Component-Oriented Programming, 9 pp.

16. Schreyjak, S., Using Components in Workflow Activities, In Proceedings of the Second and Third International Workshop on Business Objects, 1998, 12 pp.

17. Simons Anthony J H , Snoeck Monque, Hung Kitty S Y, Using design pattterns to reveal the competence of object-oriented methods in system design level, International Journal of Computer systems Science & Engineering, Vol.14, No. 6, november 1999, pp.343-352

18. Snoeck M., Dedene G. Existence Dependency: The key to semantic integrity between structural and behavioural aspects of object types, IEEE Transactions on Software Engineering , Vol. 24, No. 24, April 1998, pp.233-251

19. Snoeck M., Dedene G., Verhelst M; Depuydt A.M., Object-oriented Enterprise Modelling with MERODE, Leuven University Press, 1999

20. Snoeck M., Dedene G. Modelling the dialogue aspects of an information system, submitted for ECIS'2000

21. Vaishnavi, V., Joosten, S. & B. Kuechhler, Representing Workflow Management systems with Smart Objects, 1997, 7 pp.

22. Wolber David, Reviving Functional Decomposition in Object-oriented Design, JOOP, October 1997, pp. 31-38