CBISE'98

Synergies in a Coupled Workflow and Component-Oriented System

Stefan Schreyjak
University of Stuttgart, Germany
Stefan.Schreyjak@informatik.uni-stuttgart.de

8.–9. June 1998

Published in:
John Grundy (Ed.), *Proceedings of CBISE'98 —*
*CAiSE*98 Workshop on Component Based Information Systems Engineering*,
Working Paper 98/12, June 1998, ISSN 1170-487X
`http://www.cs.waikato.ac.nz/~jgrundy/caise98_workshop`

Abstract:
Current workflow systems allow often only modeling and execution of business processes. Application programs used in activities are integrated as black boxes. These programs can neither be created nor modified by business specialists. One must be a program developer. The use of composite applications in workflow systems can eliminate these drawbacks. These applications are built of components and are modeled in a way similar to workflows. A coupling between a workflow and a component-oriented system has advantages compared to their combination. Context-sensitive applications and more flexible workflow models are becoming possible.

Keywords: business objects, components, component-oriented system, composite applications, workflow management system

1 Introduction

To provide a better understanding of this paper, some basic terms of workflow management and component-oriented systems will be introduced.

A *workflow management system* is a software system for coordination and cooperative execution of work according to a specification (see also [14, 1, 2]). The use of a workflow system can be divided into a modeling phase, an execution phase, and an analyzing phase.

In the modeling phase, the organizational structure of an enterprise or administration and the order of all business procedures are modeled as *business processes*. The whole work is splitted into several steps, which are connected by a relationship. This chronological order is forming the process. A process step, also called *activity*, is a piece of continuous work performed by one person or machine—the *actor*. Each activity is associated with consumed and produced work objects (i.e., data and documents) and with organizational (e.g., a *role*) and technical resources (e.g., an application program) that are necessary for performance. The modeled business process is specified as a *workflow scheme* (often called a *workflow model*) in a formal textual or visual workflow language.

In the execution phase, these schemes are used for controling, supervising and recording of activities performed. The executable instance of a workflow scheme is called a *workflow instance*. A *workflow* represents a running workflow instance in the enterprise. The actor of an activity can use interactive application programs to fulfill the objective of the activity. As an alternative, manual activities without computer assistance might also be possible as well as automated activities, which do not need human interaction.

In the analyzing phase, recorded data are evaluated. The results extracted serve as a basis for improvement of workflow schemes. Passing all phases several times in a cycle, business processes are optimized little by little.

A *component-oriented system* [11] is an architecture or an implementation for a software system providing infrastructure for components. An implementation consists of an application development environment and a runtime environment for components. A *component-based system* [13] uses components only during development. In the runtime environment (i.e., the compiled application), components do not exist anymore. They are hidden from the user. The term *composite application* defines an application that is built using components as building blocks. The term is used to distinguish between composite and monolithic applications that are built conventionally.

In the following section, some problems will be described resulting from the restriction and deficient flexibility of monolithic applications that are used in workflow systems. Subsequently, a design of a system is introduced that couples a workflow and a component-oriented system. This coupled system can solve those problems. After that, the additional benefit of a tight coupling compared to a pure combination of individual systems is explained. The paper ends with a description of related architectures and a conclusion.

2 Problem Description

The engagement of current workflow systems reveals some problems linked with creation, use, and adaptation of applications. The use of component-oriented systems as application programs in activities can eliminate or at least reduce these problems [8].

2.1 Creation of Application Programs

Workflow systems support the (partial) automated handling of business processes. However, support is restricted to the process-oriented part of a business case. The function-oriented part, which comprises applications realizing business functions, is usually supposed to be given. These applications are executed in activities. If there is no suitable application for the activity while implementing a business process, the application program has to be realized in a conventional way. For example, an order is given to an external or internal software vendor. A workflow system offers usually no support for realization of application programs. Traditionally, the realization of workflows and applications are kept separate.

2.2 Adaptability Problem in Activities

Using a modeling tool, a user can easily adapt a workflow to changing business situations. However, adaptability ends at the border of activities. Programs executed in activities can only be adapted by software developers. This can be expensive and time-consuming. Component-oriented systems allow the user to adapt composite applications. Indeed, this feature is not supported very well in current systems [5].

2.3 Heterogenous System Environments

Workflow systems are often used in a very heterogenous infrastructure of computer equipment, which is grown by and by. Workflows are usually based on a platform independent specification. Nevertheless, the invoked application are often platform dependent. This problem can be solved if platform independent components are used to build composite applications.

2.4 Incomplete Reuse of Workflows

Reusing a workflow in a different enterprise requires the workflow to be adaptable by the user. Additionally, each application have to be executable on each platform used enterprise-wide. As a consequence, without the availability of adaptations and of platform independent applications, reuse of workflows is not really possible.

2.5 Unspecific Applications

In particular, if workflows are executed often, workflow systems are very useful. Therefore, applications should be accommodated very well to their operating area to achieve high productivity. However, creation of customized applications is usually expensive. Component-oriented systems allow to create and modify activity-specific programs simply and quickly. Moreover, composite applications can easily be customized to specific application domains.

3 Architecture of a Coupled System

Workflow systems and component-oriented systems have similar assignments and use similar technical approaches to facilitate development of applications. However, they differ in application domain and in magnitude of application. Workflow systems control enterprise-wide applications consisting of activities. Component-oriented systems control local applications consisting of software building blocks. This similarity can be explored for a two-way exchange of architectural concepts. A design of an architecture [9] coupling both systems is introduced in this section. This proposed architecture can solve the problems mentioned in the previous section. The specific architectures of the individual systems can be found in [2, 6]. The design concentrates on the question how to create activity-specific applications. Legacy applications can be used in the workflow system, as usual.

In the coupled system, the workflow system is responsible for the process-oriented part of the business process and coordinates the order of activities in a typically distributed, heterogenous computer environment. The component-oriented system is responsible for the function-oriented part inside an activity and coordinates working steps. The component-oriented system serves not only for execution but also for creation and modification of application programs in activities. Besides the coupling in the execution phase, there exists also a coupling in the modeling phase.

The principle for modeling workflows is adopted and applied to composite applications. By analogy with workflows, applications are divided in already existing building blocks (i.e., the function part). The blocks are connected by a modeled control flow (i.e., the process part). This approach is better than the more conventional approach to code an application in a programming language monolithically. Composite applications can be created and modified by the user himself more easily. Programming code, however, is only understood by system developers.

From the workflow system's point of view, an activity has an inner structure. Instead of being a black box, it consists of working steps connected by a modeled control flow. One could say, the component-oriented system can be seen as a simplified and specialized 'mini workflow system' inside activities. In some aspects, it provides the same functionality as a workflow system.

The component-oriented system is based on an architecture that is a variant of the compound document concept [6] and is adapted to workflow management domain. The user arranges data that represent business semantics into a uniform collection, called *composite application*. The arrangement of data is done similar to that in a compound document.

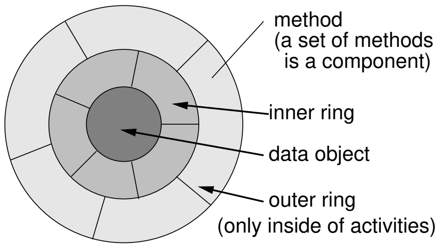

Figure 1: The structure of a business object

A composite application consists of *business objects* [6, 5] representing entities in a business domain. Each object has a specific meaning commonly used by business specialists. An example is the business object *Customer* in a composite application *Tender Management*. Business objects are divided in pure data objects and atomic or compound component objects.

The data itself or links to it are stored in *data objects*. Data objects establish a data integration level providing a uniform view on workflow-relevant data. In the example, the business object *Customer* consists of a data object that is a link to a tuple in the relation *Customers* stored in a relational database management system.

Atomic components offer common usable functions restricted exactly to one type of data object. Typical functions are creating, displaying, or editing of data. As a general rule, atomic components can only be configured but not created by users. The development of atomic components should be taken over by an independent software vendor. Thus, the vendor can hide its expert knowledge from the user. In the business object *Customer*, an atomic component can be used, for example, to exchange one tuple against another tuple of the same relation. *Compound components* are used to embed other composite applications.

Components of the inner ring (see Figure 1) are usually associated with a data object during the whole life cycle of a composite application. They shall be used to implement common functions. *Components of the outer ring* only exist inside of activities. The components are associated with the data objects of a composite application. They shall be used to realize activity-specific functions on certain data objects.

The interactive or automatic creation or modification of any data objects through a component function is called a *working step*. Inside an activity, a control flow is defined on working steps.

A workflow scheme describes a workflow separated in different aspects [1]. Thus, modifications in one aspect do not propagate inevitably into other aspects. Some independence can be achieved. This principle can be applied to the modeling of composite applications, too. Because some aspects are modeled in both systems (i.e., control and data flow), it suggests itself to couple both models in these aspects.

One or more composite applications can be associated with activities. A control flow is defined for each composite application used in an activity. Additionally, it is specified in which state a composite application must be if the activity shall be finished.

The user is able to modify a composite application on two levels. On the first level of modification, the user needs no programming skill. He or she can modify data objects by the means of components. Business objects can be configured. Even more, the arrangement of business objects in the composite application can be changed by adding new ones or deleting old ones. On the second modification level, the user can modify the control flow between business objects. Calls to component functions (working steps) can be added or deleted. Though, depending on the scope of modification, he or she needs programming skill in the (scripting) language the control flow is modeled with.

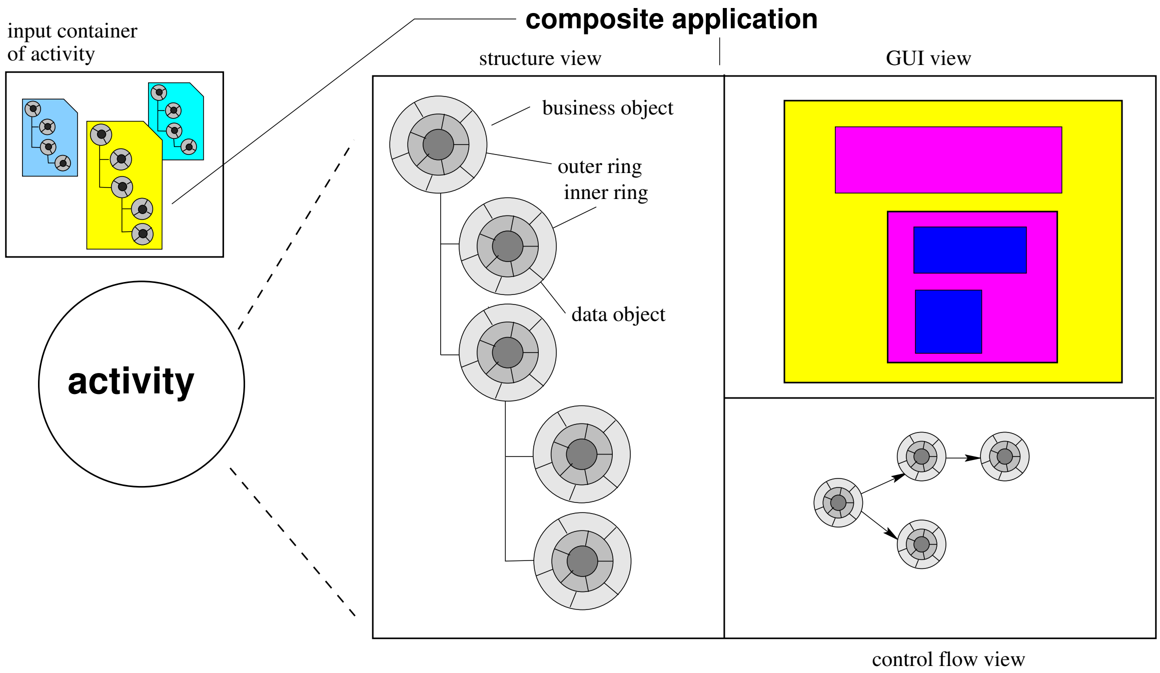

Figure 2: Sketch of a composite application in a workflow activity

An activity with a composite application is shown in Figure 2. The workflow system puts several composite application in the input container of the activity. These composite applications can be used in the activity. The structure of a composite application comprises several business objects. Inside of activity, business objects are complemented with an additional ring of application-specific components. The order of working steps on business objects is described by a specified control flow.

This control flow is used for guiding the user during his or her work.

This coupled system architecture enables the end user to create and modify whole business processes. He or she can adapt the workflow model as well as the model for the composite application according to changing business situations.

4 Synergy Effects through Coupling

The use of a coupled system has advantages compared with the combined (i.e., independent) use of individual systems. Three advantages are described in this section. The workflow system can automatically adapt composite applications according to their use. The coupling supports an enhanced communication method using events, which can be used to realize more flexible workflow control constructs. Common modeling aspects can be used to modify the workflow scheme more easily.

4.1 Flexible Automatic Adaptations

An application does usually not change its behavior or its outlook depending on the activity in which the application is invoked. Only input data differ in activities. If a user changes the configuration and so the behavior or outlook of an application, this adaptation can be stored either globally for all users or locally for himself. Different scopes of adaptation are not possible (e.g., a scope referring to the use). Although this problem is independent of the use of a component-oriented system, it is intensified by using composite applications. These applications consist of many, well configurable components. This is in contrast to current workflow systems using few, less configurable application programs.

The introduction of a *process context* allows flexible definition of adaptation scope based on the definition of organizational structure and process organization in a workflow system. Thus, an application can automatically be adapted according to its specific use in the activity. From the user's point of view, applications have a 'process consciousness': They 'know' in which context they are used and adapt automatically. Depending on its use, an application can behave and look different. Thus, the realization of *context-sensitive applications* will be possible.

A composite application allows several adaptations, for instance, setting attribute values of a component or changing the set of accessible functions. This is called *configuration*. It is possible to fade out certain data objects or components or to change the associated component of a certain data type.

The modeling tool of a workflow system must allow to define contexts in the definition of organizational structure and process organization. A process context consists of a set of activities that are associated semantically. A context is not restricted to activities of one process. It can comprise activities of several processes. For example, the activities are related because of their common application domain or use. A context can consist of activities performed by certain persons or roles, too. The affiliation to a process context can also depend on workflow-relevant application data. A workflow system can manage several contexts. An activity can be a member of one or more contexts. Therefore, a relationship on contexts must exist that defines an order in which the adaptations of the composite applications have to be done. A specific part of the workflow system is responsible for the adaptation of a composite application. Before an activity is started, the adaptation has to be done in the activity. In addition, a database for configuration data is necessary.

Process contexts can be applied diversely. For instance, a context can supply pre-selected data values in input masks or it can restrict the history of input values to input made in earlier workflow instances of the same workflow scheme. By defining a workflow as *uses-foreign-language*, the user interface can be configured automatically for a certain language by the means of a context. Composite applications can show different views on included or referenced data depending on the person or role (e.g., an expert) who is working with the application. Certain properties (e.g., protocoling all actions) can flexibly and automatically be switched on or off via the context.

4.2 More Flexible Control in Workflows

Typically, an activity is integrated in a workflow system using a model called *black box integration*. The model is restricted to invocating an application, supplying or transferring parametric data, and waiting for completion of the application. An application can be wrapped with an *envelope* [12] to make the invocation interface of application compatible to the interface of the workflow system. There is no need for modifying source code. From the workflow modeler's point of view, a conventional application is built monolithically—even if source code exists. Internal states or occurred events of an application cannot be made public without having high programming skill.

In composite applications, the user can modify some aspects, in particular control flow and communication. The boundary between black box and white box integration is being blurred. So to say, a composite application consists of 'black' components and 'white' control flow.

Cooperation between the workflow and component-oriented system can be facilitated through the mutual exchange of events. This communication method can be used for more fine-grained synchronization or for externalization of application states. Therefore, descriptive control flow constructs [2] can be implemented more easily. That helps to model the control flow of a workflow more accurately.

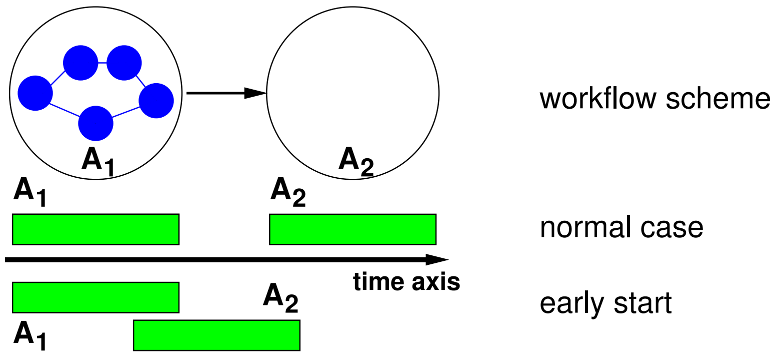

Figure 3: Early start of A_2

For example, the control construct *early-start* allows to start a succeeding activity A_2 before the preceeding activity A_1 is finished. In some situations, this construct can increase parallelism. A_1 possesses an inner structure—the process consisting of working steps. If a certain state in this inner process is reached, the actor of A_2 can start A_2 earlier by its request. If the workflow system should support this, it must be informed about the progress of processing in the activity.

Flexible communication methods are important for the interoperation between workflow and component-oriented system. For example, the realization of interrupting, aborting, or delegating activities is easier. *Delegation* is the migration of an activity to another user. In current workflow systems, delegation can often only be done when the activity has not been started. Delegating while the activity is being performed requires sophisticated communication methods. Thus, in a coupled system, individual working steps can be executed by another user.

The communication methods can also be used to realize a more fine-grained version management. In long-living activities, several versions can be delivered up the workflow system which can associate these versions directly with the workflow. If an application had used an individual version management inside the activity, it should had associated the versions to the workflow explicitly.

4.3 Shifting of Application Parts

Whenever business increases, new workers are engaged. If the new workers have the same assignments as the old ones, the workflow system can allocate the work load to the workers fairly. If the assignment is different, the old worker's assignment has to be divided and a part has to be assigned to the new worker. This partition of work assignments requires that an activity must be divided, too. If there is an activity-specific application used, it has also to be splitted and parts have to be shifted into a new application of a new activity.

Shifting is a modification in the workflow scheme in which parts of an application are separated and merged into another application. In the introduced architecture, an application part consists of working steps associated with control flow.

The architecture supports this kind of modification well because the coupled modeling in the modeling phase can be used. A composite application is structured in a collection of business objects and control flow. These structure allows identification of application parts. Therefore, even unskilled user can separate parts and insert them elsewhere in other applications. The workflow system facilitates the performance of shifting because it can use information about the whole process to reveal consistency problems. It can point the modeler to activities or applications in which 'holes' have occurred. The modeler has to 'fill' these holes—by doing some further modifications. In the case of inserting, the workflow system can also warn of problems, like missing data objects.

5 Related Work

Besides the introduced architecture of a coupled system, some more architectures exist that support inner structure in activities. These are the integration approach and the distributed workflow system approach.

In the integration approach, the inner structure is modeled as a subordinated workflow. Each activity has exactly one component associated [3, 10]. This approach rises sharply the number of activities that have to be performed by the workflow engine. It is not likely that a centralized workflow engine can cope with this additional load and deliver the required performance for interactive applications. Therefore, scalability of this approach is bad.

As an alternative, workflow systems can be realized in several more decentralized ways [4]. An architecture can consist of several interoperating workflow engines whose realizations are central. The coupled system is also an architecture in which workflow systems interoperate. However, the systems are not equivalent but specialized for a certain purpose. Another architecture, a fully distributed workflow system [7], has no centralized engine anymore. Each activity has its own part of control functions. The distributed architecture matches the inherent distributed character of the execution phase very well. But the modeling and analyzing phase have a inherent centralized character. The realization of these phases is becoming more expensive.

To be comparable with the introduced system architecture, all approaches must be enhanced with methods to adapt a workflow. This can also be expensive if the workflow system is mainly distributed.

6 Conclusion

The use of a component-oriented system for creation of applications, which are invoked in workflow activities, increases flexibility of a workflow system. The basic principle for user's self-organization in business processes is going to be real. The end user can create business-wide applications and he or she can modify the whole business process according to changes in business situations. The end user is not restricted to modifications of the pure workflow model anymore (i.e., the workflow without the applications). The user's working method can be adapted and arranged to his or her own needs more efficiently. The coupling instead of simply combining those two systems increases the benefit even more. Synergy effects are becoming possible that are not possible in individual systems.

The coupling and the synergy show diverse forms: A process context can be used to automatically adapt composite applications to their specific application domain context-sensitively. According to its intented purpose, an application can react differently. Events occurring in composite applications can trigger events on the workflow level and vice versa. Synchronization mechanisms are becoming more flexible. Thus, real business processes can be modeled more accurately in workflow systems. Certain aspects in composite applications and in workflows can be modeled in common. This simplifies consistent modifications, like shifting of application parts.

References

[1] Stefan Jablonski. Workflow-Management-Systeme: Motivation, Modellierung, Architektur. *Informatik Spektrum*, 18:13–24, 1995.

[2] Stefan Jablonski, Markus Böhm, and Wolfgang Schulze. *Workflow-Management: Entwicklung von Anwendungen und Systemen*. dpunkt Verlag, 1997.

[3] Frank Leymann. Workflows Make Objects Really Useful. In *Proceedings of the 6th International Workshop on High Performance Transaction Systems (HPTS)*, September 1995. http://www3.hursley.ibm.com/hpts95/proc95.htm.

[4] J. A. Miller, A. P. Sheth, K. J. Kochut, and X. Wang. Corba–based Run–time Architecture for Workflow Management Systems. *Journal of Database Management*, 7, 1996. Special Issue on Multidatabase.

[5] OMG. Common Facility RFP-4 — Common Business Objects and Business Object Facility. Technical report, Object Management Group, 1996.

[6] Robert Orfali, Dan Harkey, and Jeri Edwards. *The Essential Distributed Objects Survival Guide*. Wiley & Sons, New York, NY [u.a.], 1996.

[7] Rainer Schmidt. Component-based Systems, Composite Applications and Workflow Management. In *Proceedings of Foundations of Component-based Systems Workshop*, pages 206–214, September 1997.

[8] Stefan Schreyjak. Coupling of Workflow and Component-oriented Systems. In Wolfgang Weck, Jan Bosch, and Clemens Szyperski, editors, *Proceedings of the* 2^{nd} *International Workshop on Component-Oriented Programming (WCOP'97)*, TUCS General Publications No 5, pages 77–86. Turku Centre for Computer Science, September 1997.

[9] Stefan Schreyjak. Using Components in Workflow Activities. In Jeff Sutherland and Dilip Patel, editors, *Proceedings of the* 2^{nd} *and* 3^{rd} *International Workshop on Business Objects (at OOPSLA)*, Springer Verlag, 1998. to appear.

[10] Stefan Schreyjak and Hubert Bildstein. *Beschreibung des prototypisch implementieren Workflowsystems Surro*. Universität Stuttgart, Software–Labor. Fakultätsbericht Nr. 1996/19.

[11] Clemens Szyperski. Independently Extensible Systems — Software Engineering Potential and Challenges. In *Australian Computer Science Communications*, 18(1):203–212, February 1996.

[12] Giuseppe Valetto and Gail E. Kaiser. Enveloping Sophisticated Tools into Process-centered Environments. *Journal of Automated Software Engineering*, 3:309–345, 1996.

[13] Peter Wegener. Concepts and Paradigms of Object–oriented Programming. *OOPS Messenger*, 1(1):8–87, 1990.

[14] Workflow Management Coalition. *The Workflow Referenz Model*. Technical report, November 1994. Document Number WFMC-TC00-1003, Issue 1.1, http://www.wfmc.org/wfmc/DOCS/refmodel/rmv1-16.html.