Acta Scientiarum 22(5):1311-1319, 2000.
ISSN 1415-6814.

Application of an object-oriented framework for task scheduling in an ExPSEE environment

Itana Maria de Souza Gimenes[1]* and Sergio A. Tanaka[2]

[1]*Departamento de Informática, Universidade Estadual de Maringá, Av. Colombo, 5790, 87020-900, Maringá-Paraná, Brazil.*
[2]*Cesulon, Av. JK 1626, Centro 86020-000 Londrina-Paraná, Brazil. *Author for correspondence. e-mail: itana@din.uem.br*

ABSTRACT. The application of an object-oriented framework for task scheduling in the ExPSEE Environment is provided. ExPSEE is an experimental Process-centred Software Engineering Environment. However, this framework can be used in other domains such as Workflow Management Systems and Project Management Systems. The task scheduling framework was based on both current methods for the development of frameworks and on an existing architectural pattern for process managers. A prototype of the framework was developed using the Java Language. It shows the experience of extracting a framework from well-known applications that can be reused in practical domains. The research in current paper contributes in the production of a framework and gives insights in the application of novel techniques towards the development of frameworks.

Key words: frameworks, software engineering environment, software process, workflow, task scheduling.

RESUMO. Aplicação de um framework orientado a objetos para escalonamento de tarefas no ambiente ExPSEE. Este artigo apresenta a aplicação de um framework orientado a objetos para escalonamento de tarefas no ambiente ExPSEE. ExPSEE é um ambiente experimental de engenharia de software orientado a processos, no entanto, o framework também pode ser utilizado em outros domínios tais como sistemas de gerenciamento de workflow e sistemas de gerenciamento de projetos. O framework foi concebido com base em métodos atuais para desenvolvimento de frameworks e em um padrão existente para gerenciadores de processos. Um protótipo do framework foi desenvolvido usando a linguagem Java. O trabalho desenvolvido mostra a experiência de extração de um framework a partir de aplicações conhecidas e que pode ser reutilizado em domínios de aplicação prática. Assim, o presente trabalho contribui tanto na produção de um framework quanto na geração de conhecimento sobre a aplicação de técnicas inovadoras de desenvolvimento de frameworks.

Palavras-chave: frameworks, ambiente de engenharia de software, processo de software, workflow, escalonamento de tarefas.

The objectives of software engineering are centred both on improving the quality of the software process and the quality of the software product itself. Techniques for increasing productivity and reducing costs and production efforts must also be taken into account. Frameworks and components reuse is an important issue in this context. It allows the development of software based on components, frameworks and models (templates) that are already well specified and tested.

Software engineering techniques used in the definition and application of software architectures, frameworks, patterns and components are rapidly evolving. These concepts have already reached the commercial organisations triggering a new market of products.

An object-oriented framework is generally characterized as a set of abstract and concrete classes, plus their collaboration relationships, which offer an implementation scheme for applications (Lewis *et al.* 1995). The Catalysis method (D'Souza and Wills, 1999) proposes a wider concept of frameworks, called model frameworks. Model frameworks are designed at a higher level of abstraction establishing a generic scheme that can be imported, at the design level, with substitutions and extensions in order to generate specific applications. This paper follows this approach and presents a framework for PSEE (Process-centred Software Engineering Environments) and WfMS (Workflow Management Systems).

A software process is composed of an ordered set of tasks for the development and maintenance of software products. PSEE are environments that support the management and automation of software processes.

Workflow technology (Jablonski and Bussler, 1996) meets the current needs of organizations as the reengineering of legacy processes and the modeling and automation of business processes, supported by workflow systems, are means to improve the productivity and the quality of processes and products. Moreover, workflow systems allow rapid development and modification of systems to comply with the transient and unexpected variations of the business environment.

Workflow systems are applications supported by WfMS. These systems support definition, management and execution of workflows. WfMS interpret process definitions, interact with the participant users (the human agents), and, when necessary, they invoke tools and applications to execute parts of the workflow (Workflow, 1999), (Workflow, 1995). WfMS and PSEE have many similar features. The software process can be seen as a workflow for the production of software.

This paper presents the application of a framework for task scheduling to PSEE. The ExPSEE is the Experimental Process-Centred Software Engineering Environment where the application of the proposed framework took place. However, the framework can also be reused in applications such as WfMS or any application that involves some form of task management (ex. building or course management). The framework was based on both current methods for the development of frameworks and on an existing architectural pattern for process managers. A prototype of the framework was developed using the Java Language. The complete specification of the framework was presented in (Gimenes *et al.*, 2000) (Tanaka, 2000). Section 2 presents the domain analysis of the proposed framework. Section 3 and 4 describe the definition and the design of the framework for task scheduling respectively. Section 5 discusses the connection of the framework in ExPSEE application. Section 6 discusses the prototyping and validation of the framework. Finally, section 7, deals with the conclusions and future work.

Domain analysis of the proposed framework

The domain analysis took a similar approach to the Example Driven Project (Johnson, 1996) in which frameworks are identified using application examples and are generalised in an iterative process. The main application examples explored for domain analysis were WfMS and PSEE. Although these applications are distinct, they have many common elements. Thus, the objective of the domain analysis is to identity these common elements as they constitute potential frameworks.

The process manager pattern. The Process Manager is an architectural pattern for definition of PSEE process managers (Gimenes *et al.*, 1999a). It was developed from studies of existing environments (Finkelstein, 1994) and from experiences obtained in the development of the ExPSEE (Experimental Process-centred Software Engineering Environment) project (funded by CNPq - a Brazilian research funding agency). The pattern is based on a process model that allows the definition and reuse of process architectures. It is described according to (Buschmann *et al.*, 1996), using UML (Unified Modelling Language) (Rumbaugh J. *et al*, 1999) (Rational, 2000) to represent the diagrams that compose it. Figure 1 shows the main structure of the pattern through its class diagram.

The top part of the diagram represents the classes correspondent to the modules of the process manager. The central and inferior parts of the diagram were defined to support process architecture reuse. The central part involves the definition of the process architecture and the object types related to it, whereas the inferior part represents the software process instantiation, according to the defined architecture, and its related objects.

The WfMC reference model. WfMS are based on the WfMC (Workflow Management Coalition) reference models (Workflow, 1995). They include a generic architecture for WfMS, as shown in Figure 2, and a reference model which encompasses 5 categories of interoperability and communication standards.

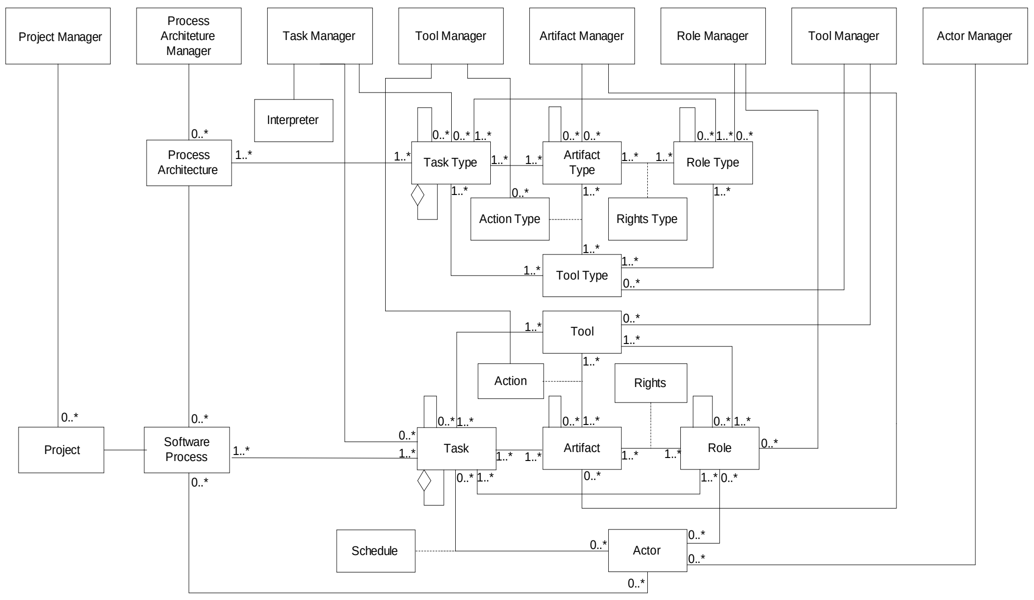


Figure 1. Class diagram of the Process Manager pattern (Gimenes *et al*., 1999a)

This architecture contains the main components and interfaces that a WfMS should have in order to allow interoperability of sub-products from different suppliers, such as process definition tools.

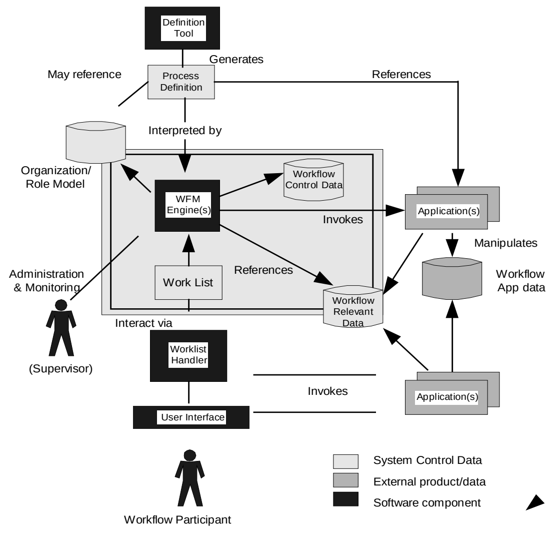


Figure 2. Generic architecture for a WfMS (Workflow, 1995)

Comparative synthesis. The class diagram of the Process Manager, shown in Figure 1, is more detailed as it was taken from the specification of the ExPSEE (Gimenes *et al.*, 1999a), (Gimenes *et al.*, 1999b). In contrast, the models of the WfMC are generic architectures that represent a reference model composed of large and abstract blocks.

Table 1 summarises the comparison undertaken between WfMS and PSEE, based on the WfMC generic architecture and the Process Manager pattern. The table rows indicate the equivalence between the common elements of both models. The number of common elements and functionalities of these systems provide evidences that the Process Manager pattern can be refined in order to achieve small frameworks that can be used in WfMS, PSEE and even extended for related applications. The identification of these frameworks represents a significant contribution to facilitate the development of this category of systems.

According to the Taligent approach (Taligent, 1996) for framework development, small and more specific frameworks for each main function of an application or domain can be produced based on a more generic framework. In the case of the process management domain dealt with in this paper, we may have frameworks for task management, artifact management, actor management and so on. These smaller frameworks have more chances to be reused in other contexts. This paper defines a framework for task scheduling with this feature. It is an important contribution as it explores the process of identification and generalisation of frameworks aiming at increasing their reuse potential.

The detailed comparison undertaken between the WfMS and the PSEE components indicates that the Worklist Handler of the WfMC generic architecture and the Task Manager of the Process Manager pattern have great similarity with regard to the functionality for the management and the execution of tasks. Thus, we have focussed our attention on the development of a framework for task scheduling as described in the following sections.

Table 1. Comparison between WfMC reference models and the Process Manager pattern

WfMC Generic Architecture Elements	Pattern Structure Elements
WfMS	Process Manager
Process Definition Tools	Process Architecture Manager
Worklist Handler	Task Manager
Workflow Control Data and Relevant Workflow Data	Artifact Manager
Applications	Tool Manager
Workflow Enactment Services	Tool Manager
Administration and Monitoring Tools	Actor Manager
Organisation Role Model	Role Manager

Definition of the task scheduling framework

The domain analysis showed that the structure of the Process Manager pattern contains many of the types, actions and collaborations that constitute the framework for task scheduling. Thus, the pattern structure was chosen as a reference for the design of the framework.

After the domain analysis, a package partitioning based on the Catalysis method (D'Souza and Wills, 1999) was carried out. Other methods for framework development also influenced the framework development, such as Taligent (Taligent, 1996). This approach is interesting, intuitive and emphasises the idea of designing small frameworks. However, a consistent notation does not support it. Catalysis proved to be better defined and more complete. The Catalysis method uses UML, and its package partitioning and model framework approach are key techniques in the identification and design of frameworks. Further, the method takes into account the reuse of patterns, the extensibility of diagrams and the interactions between diagrams.

The partitioning of the Process Manager structure was based on the class diagram presented in Figure 1 and the use cases developed within the ExPSEE project. The package separates the work in areas that can be treated individually with explicit dependencies. The package partitioning helps control the propagation of modifications, allows traceability and reduces maintenance costs. The contents of the packages represent the business rules that give a general definition of the target package. In our work we have mainly used the following partitioning approaches of Catalysis: vertical slices, horizontal slices and different domains.

The first step of the package partitioning is to analyse the existing dependencies in the classes (types, in Catalysis) diagram. The next step consists of partitioning the packages keeping a structure that facilitates and reduces the number of package importation.

Figure 3 presents the vertical slices, at a high level, of the Process Manager obtained by partitioning the system according to the actions undertaken by the main agents which interact with the environment. The agents were derived from the general use cases identified in the ExPSEE project. The vertical slices are Defining Process Architecture, Instantiating Process Architecture, Defining and Allocating Resources and Scheduling Tasks.

The vertical slices diagram also contains a basic package called Managing the Elements of the Process, which contains the managers of the basic elements of the PSEE. This package is imported by the others throughout the horizontal layers. There are two intermediate horizontal layers: the Managing Process Architecture and the Managing Projects.

Throughout the partitioning process we could clearly isolate the potential for reuse of the framework for task scheduling from the package called Scheduling Tasks, thus, identifying its types, actions and collaborations. The model framework proposed represents the scheduling of tasks used in applications such as PSEE, WfMS and project management systems.

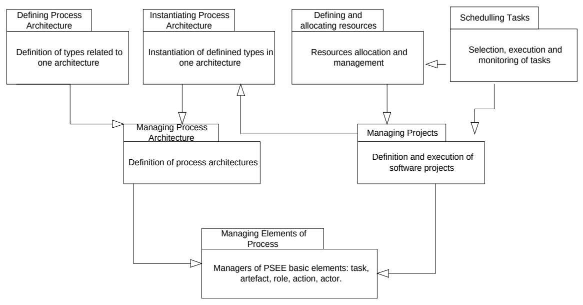

Defining Process Architecture
Definition of types related to one architecture
Instantiating Process Architecture
Instantiation of definined types in one architecture
Defining and allocating resources
Resources allocation and management
Schedulling Tasks
Selection, execution and monitoring of tasks
Managing Process Architecture
Definition of process architectures
Managing Projects
Definition and execution of software projects
Managing Elements of Process
Managers of PSEE basic elements: task, artefact, role, action, actor.

Figure 3. Vertical slices from a common base of the Process Manager

The design of the framework for task scheduling

This section presents the use cases, the design of the framework, the model of its generic package and the application of the framework in different domains.

Use cases. The use cases are presented in two figures, followed by their respective description tables, to facilitate their explanation. The first set of use cases is presented in Figure 4 and its description in Table 2. This use case has a main actor, the Project Manager, who interacts directly with the task scheduling framework.

The stereotyped relationships <<include>> and <<extend>>, defined in UML, and also considered in (D'Souza and Wills, 1999) (Schneider, 1998) are applied to the use cases. The <<extend>> is used to represent the dependency relationship between Schedule Tasks and Visualise Scheduled Tasks. The <<include>> is applied to represent the fragmentation of the use cases. For instance, Visualise Scheduled Tasks is split into Postpone Task, Update Status, Cancel Task and Set Progression Rate.

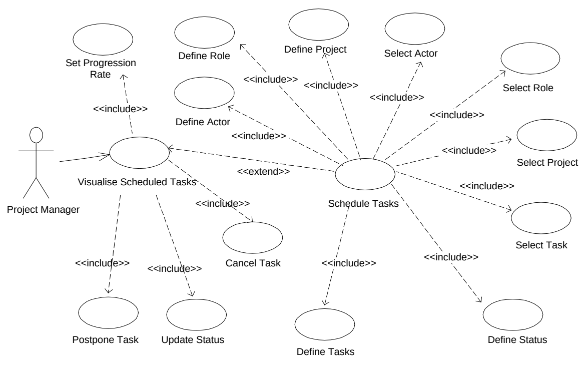


Figure 4. Use case diagram for task scheduling

Table 2. Description of the use case for task scheduling presented in Figure 4

Use Case	Who starts the action	Description
Visualise Scheduled Tasks	Project Manager	Visualisation of all scheduled tasks.
Cancel Task	Project Manager	Cancellation of previously defined tasks.
Update Status	Project Manager	Updating of the task status.
Postpone Tasks	Project Manager	Postponing the initial or final date of a task.
Set Progression Rate	Project Manager	Set the progression rate for a certain task, ex.: (0%) allocated, (25%, 50%, 75%) executed.
Schedule Tasks	Project Manager	Scheduling of a task to an actor who plays a role in a project.
Define Actor	Project Manager	Definition of a new actor.
Define Role	Project Manager	Definition of a new role.
Define Project	Project Manager	Definition of a new project.
Define Task	Project Manager	Definition of a new task.
Define Status	Project Manager	Set the status of a certain task, ex.: allocated, ready, in execution, stopped or finished.
Select Actor	Project Manager	Selection of a previously

Use Case	Who starts the action	Description
		registered actor.
Select Role	Project Manager	Selection of a previously registered role.
Select Project	Project Manager	Selection of a previously registered project.
Select Task	Project Manager	Selection of a previously registered task.

Figure 5, followed by Table 3, describes the second use case diagram of the framework for task scheduling. Besides the actor project manager, this diagram has the actor software engineer who can only access some of the funcionalities presented in the use case Visualise Personal Schedule.

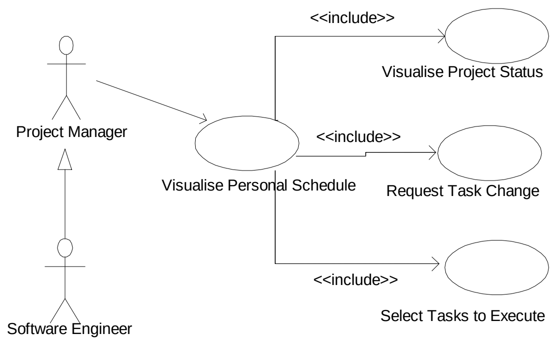


Figure 5. Use case diagram for personal task visualization

Table 3. Description of the use case diagram for Individual Task Visualisation

Use Case	Who starts the action	Description
Visualise Personal Schedule	Project Manager or Software Engineer	Visualisation of all tasks scheduled for an actor who plays a role in a certain project.
Visualise Project Status	Project Manager or Software Engineer	Visualisation of the status of a certain project including number of tasks of the projects, number of completed tasks and expected project conclusion date.
Request Task Change	Project Manager or Software Engineer	Requisition of a task change from a certain actor.
Select Tasks to Execute	Project Manager or Software Engineer	Selection of the tasks to be executed.

The model framework. According to the Catalysis method, not only pieces of code can be reused but also specifications and designs. The set of types, relationships and constraints that are specified within a package can be seen as a framework. Frameworks are represented by a generic package called model framework or template package. The framework defined in this paper is a model framework.

The framework for task scheduling is shown in Figure 6. The package diagram of the model framework was developed from the use cases described in the previous section and the types, actions and collaborations derived from the Process Manager pattern. The figure shows the interaction between the use cases and the package, a characteristic of the Catalysis model. A type defines the object by specifying its externally visible behaviour. The concept of type in Catalysis is different from the class one that describes the implementation of an object. A type is more abstract and does not prescribe an implementation.

The types with names written in brackets are defined as placeholders. These types can be substituted in the specific application. The concept is similar to the extensibility of classes of the object-oriented paradigm. There are also nonplaceholders types, such as the FrPassword which is used by the <FrIntSchedule>, represented by a dotted line. The type <FrIntSchedule> (the Schedule Interface) is related to <FrSchedule> by an aggregation. The <FrSchedule> is an association between <FrActor> and <FrTask>. It has attributes and methods through which it can reference the types <FrActor>, <FrRole>, <FrProject> and <FrTask>. The <FrTask> is related to itself by an aggregation, indicating that one task may be composed of several tasks.

The use cases are represented by ellipses that define the actor responsible for the action execution. The use case diagrams as described in Figures 4 and 5 represent the detailed actions. The main actions of the use cases are represented in Figure 6: Visualise Personal Schedule, Schedule Task and Visualise Scheduled Tasks.

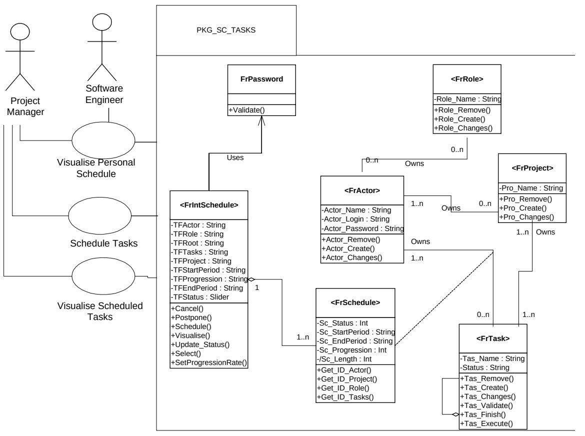


Figure 6. Generic framework package diagram for the task scheduling

Figure 7 shows the application of the framework for task scheduling in the ExPSEE context, according to the Catalysis notation. In this figure we can see the application of the types defined as placeholders. The application of the framework is carried out by importing the placeholders with substitutions. This is represented by labelled arrows from the package to each substitution. It can be observed that the placeholders are substituted by the correspondent classes of the ExPSEE Process Manager presented in Figure 1.

The framework for task scheduling can be reused in several domains. Figure 8a shows the application of the framework in building management, whereas Figure 8b shows the application of the framework in the control of subjects offered in a course. In the first application, the types <FrProject>, <FrRole>, <FrTask>, <FrActor>, <FrSchedule> and <FrIntSchedule> are replaced by the classes Building, Role, Task, Person, Schedule and BuildingSchedule. In the course control application the same types are substituted by Course, Teacher, Subjects, Person, Schedule and SubjectSchedule.

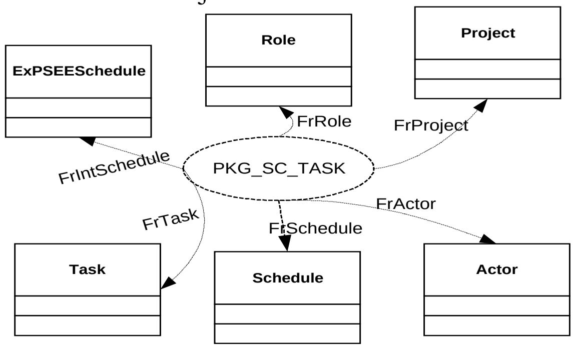


Figure 7. Application of the Task Scheduling Framework in the ExPSEE context

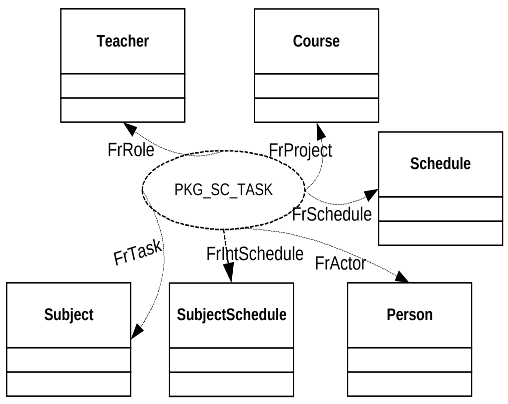


Figure 8a. Application of the framework in the building management domain

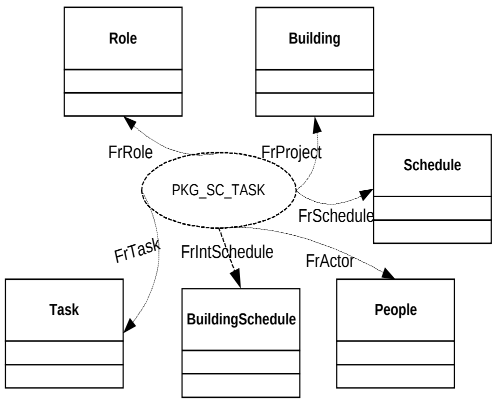


Figure 8b. Application of the framework in the scheduling of subjects in course management applications

The application of a model framework is further unfolded to represent the complete model of the specific application. Unfolding includes the substitutions in the target context as well as additional implementation classes.

A case study was carried out in which the framework for task scheduling was unfolded down to the level of implementation, taking into consideration the ExPSEE context. Besides the types presented in Figure 6, unfolding, shown in Figure 9, contains implementation classes. For instance, the class FrSchedule_Task contains the actions for personal scheduling of a certain actor, role, task and project and allows the insertion of new registers. The class FrVisualise represents the visualisation of the tasks from a certain actor who works in a given project playing an established role. The class Request was added to represent the requests made by the software engineers to the project manager, informing the problem description.

The class ExPSEESchedule is the result of the substitution and specialisation of the type <FrIntSchedule>. This class represents the visual interface through which the project manager and the software engineers visualise all the scheduled tasks.

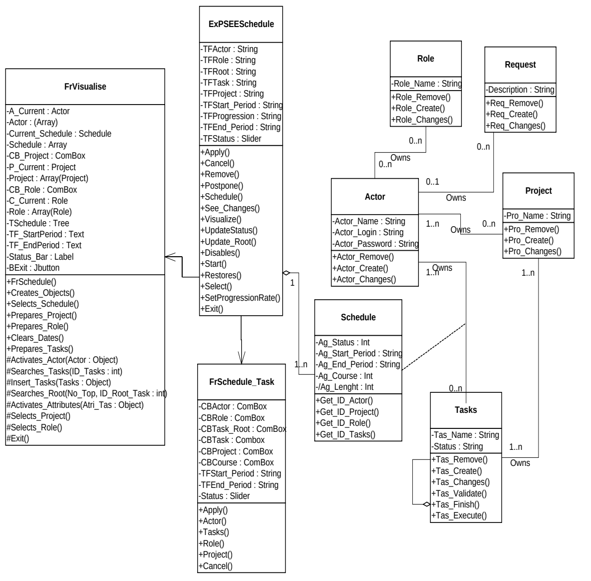


Figure 9. Unfolding of the application of the task scheduling framework in the ExPSEE context

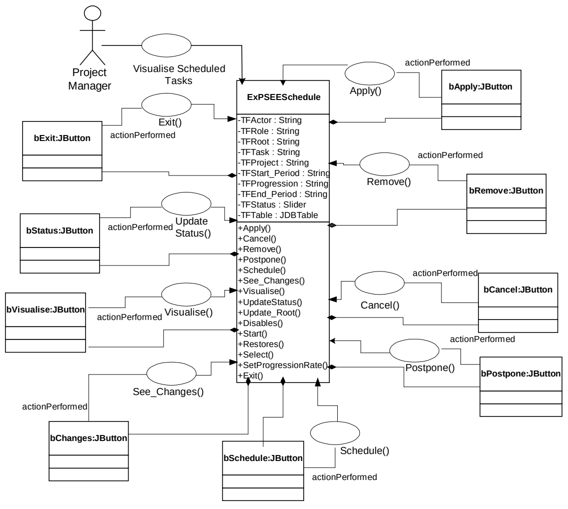


Figure 10. Specification of the task scheduling in the ExPSEE

Connecting frameworks to applications

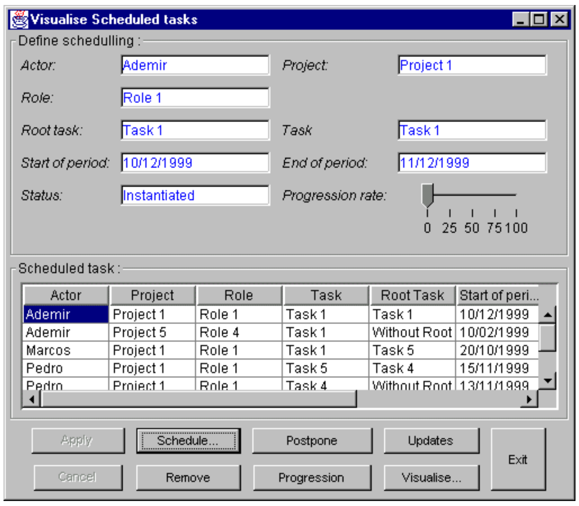
According to Catalysis (D'Souza and Wills, 1999), a component framework establishes a collaboration within which all its elements are specified as types. In order to use the framework, the elements of connection (plug ins and plug points) must be plugged in the applications. The implementation of the connections may have even more functionalities than the specified ones due to their generalization.

The most usual form of connection provided by object-oriented programming languages is the class inheritance approach. The superclasses implement skeleton of applications within the framework. Methods of these superclasses call the operations defined in the subclasses to accomplish the connection.

Thus, a way of connecting the task scheduling framework to applications is by importing it by using the class inheritance approach. The application must also be prepared to allow the framework connection and thus makes plug points available. A framework may have different interfaces (front ends) according to its different users. This view is shown in Figure 10 and takes into account the use of the framework in the ExPSEE context. In Figure 10, the letter b (lower case) represents a button that triggers the action represented by the ellipses.

Prototyping and validating the framework

A prototype was developed to validate the framework for task scheduling. The prototype was implemented in the Java Language and used Microsoft Access 97 (Microsoft, 1997) to manage the database. The task scheduling prototype contains the visual interfaces through which the actors may select the tasks to be executed and to obtain information about their tasks. Figure 11 shows the interface Visualise Scheduled Tasks. Interface also shows attributes such as start and end date, root task, progression rate and task status. The interface also provides buttons to start actions such as apply, cancel, schedule, remove, postpone, visualise status and updates, and exit. These actions correspond to the use cases in Figure 4. When the option Visualise is chosen, a password is requested from the actor. Afterwards, the interface shows the actor schedule that contains the tasks that he/she can execute. This corresponds to the software engineer use cases presented in Figure 5.



Actor:	Ademir	Project:	Project 1
Role:	Role 1		
Root task:	Task 1	Task	Task 1
Start of period:	10/12/1999	End of period:	11/12/1999
Status:	Instantiated	Progression rate:	0 25 50 75 100

Actor	Project	Role	Task	Root Task	Start of peri...
Ademir	Project 1	Role 1	Task 1	Task 1	10/12/1999
Ademir	Project 5	Role 4	Task 1	Without Root	10/02/1999
Marcos	Project 1	Role 1	Task 1	Task 5	20/10/1999
Pedro	Project 1	Role 1	Task 5	Task 4	15/11/1999
Pedro	Project 1	Role 1	Task 4	Without Root	13/11/1999

Figure 11. Interface of the framework for task scheduling

The connection of a framework within the specific applications is made through plug points. In the prototype, the plug ins were implemented by class extensions.

The development of the task scheduling framework in the ExPSEE context required a re-evaluation of the implementation of the ExPSEE so that its architecture could become more flexible and allow the reuse of other frameworks.

Discussion

Research work in current paper shows the similarities between the reference models of WfMC and PSEE. The comparative analysis of these systems indicates that it is possible to define generic frameworks that can be reused in these domains. Thus, this paper proposes a framework for task scheduling, which can be used in these domains or related applications such as task scheduling in building or course management. It also shows the process of defining the framework. This paper shows the application of the proposed framework to PSEEs. In particular, the framework was applied to the ExPSEE environment.

The Process Manager pattern was used as a basis to derive the proposed framework. The main development method was Catalysis. The method was used due to its emphasis on factoring and composing frameworks and components. The package partitioning of Catalysis and the model framework proved to be very useful. The package partitioning is important to discuss how to refactor the system while respecting the users category views and structuring the system into horizontal layers. The model framework approach enables the design of framework to abstract away from any implementation details. Further, the model framework can be used as a design template to generate components for more specific applications. This facilitates the production of components for any domain.

The use of an architectural pattern was important since it gave the main structure from which the types, actions and collaborations of the framework were derived. Architectural patterns provide generic structures that can be used to derive small and more specific frameworks. These smaller frameworks are usually less complex and easier to understand. Therefore, they have higher reuse potentiality.

The development of frameworks requires time, experience and techniques that range from the domain analysis to their implementation and use. In the development of the framework for task scheduling the experience in the previous development of the ExPSEE and the Process Manager pattern proved to be a key issue.

The application of the framework for task scheduling to the ExPSEE environment made it possible to validate and test the framework. The application forced a re-evaluation of the implementation of ExPSEE so that its architecture became more flexible and permitted the reutilization of other frameworks.

The possibility of reusing this framework in several domains was emphasised in sections 2 and 4. Further work will be concerned with applications of task scheduling in WfMS to manage courses. This work will take place in the context of the Tapejara Project (Oliveira *et al*., 1998) (a cooperative project funded by CNPq/Protem-CC) which aims at developing an environment for distance learning, training and education supported by the workflow technology. In this environment the task scheduling is a key issue to obtain successful results of the courses. This work will give an even deeper analysis of the similarities and adaptations necessary to the application of the framework proposed in this paper.

References

Buschmann, F. *A System of Patterns*. Pattern – Oriented Software Architecture. New York: John Wiley, 1996.

D'Souza, D.; Wills, A. *Objects, Components and Frameworks with UML* - The Catalysis Approach, USA: Addison-Wesley, 1999.

Finkelstein, A.; Kramer, J.; Nuseibeh, B. (Ed.). *Software Process Modelling and Technology*, England: John Wiley, 1994.

Gimenes, I.M.S.; Weiss, G.M.; Huzita, G.H.M. Um padrão para definição de um gerenciador de processos de software, In: WORKSHOP IBERO AMERICANO DE ENGENHARIA DE REQUISITOS E AMBIENTES DE SOFTWARE, 2., 1999. *Memorias ...* Cartago: Cit, 1999. p. 36-46.

Gimenes, I.M.S.; Huzita, G.H.M.; Carniello, A.; Fantinato, N. ExPSEE. An Experimental Process Centred Software Engineering Environment. Maringá: UEM/CTC/DIN, 1999.

Gimenes, I.M.S.; Tanaka, S.A.; Palazzo, J.P. An object-oriented framework for task scheduling. In: TOOLS EUROPE 2000, 2000, Mont St Michel - France. Tools 33 Technology of Object-oriented Language and Systems. Los Alamitos, USA: IEEE Computer Society Press, 2000. v.1. p.383-394.

Jablonski, S.; Bussler C. Workflow Management Modeling Concepts, Architecture and Implementation, England: Thomson Computer Press, 1996.

Johnson, R. How to Develop Frameworks, In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, ECOOP, 1996, Linz, AT. *Tutorial Notes ...* [Linz:s.n.], 1996.

Lewis, T. *Object Oriented Application Frameworks*, USA: Manning, USA, 1995.

Microsoft Office. Microsoft Access. User Guide. *Relational Database Managerial System for Windows.* 1997.

Oliveira, J.P.; Nicolao, M.; Edelweiss, N. Conceptual Workflow Modelling for Remote Courses, In: IFIP WORLD COMPUTER CONGRESS, TELETEACHING '98 DISTANCE LEARNING, TRAINING AND EDUCATION, 15., 1998. *Proceedings ...* Vienna, 1998. p. 789-797.

Pree, W. Design Patterns for Object-Oriented Software Development. USA: Addison-Wesley, 1995.

Rumbaugh, J.; Jacobson, I.; Booch, G. T*he Unified Language Reference Manual.* USA: Addison-Wesley, 1999.

Rational Software Corporation, *UML Notation Guide*, Available at: http://www.rational.com/uml/html/notation, 2000.

Scheider, G.; wiinters J.P. *Applying Use Cases a Practical Guide*, USA: Addison-Wesley, 1998.

Taligent White Paper, *Building Object-Oriented Frameworks*, Taligent, Inc, Apple Computer, Inc, IBM Corporation and Hewlett-Packard Company, 1996. Disponível em: http://www-4.ibm.com/software/ace/taligent

Tanaka, S. *Um Framework de Agenda de Tarefas para Gerenciadores de Processos.* Porto Alegre, 2000. (Master's Thesis in Computer Science) - PPGCC, Institute of Information, UFRGS, 2000.

Workflow Management Coalition, Terminology & Glossary, Technical Report TC-1011, 1999. Disponível em: http://www.wfmc.org

Workflow Management Coalition, The Workflow Reference Model, Technical Report TC-1003, 1995. Disponível em: http://www.wfmc.org.

Received on October 17, 2000.
Accepted on November 27, 2000.